VNIVERSITAT DE VALÈNCIA

Universidad de Valencia
Intelligent Data Analysis Laboratory (IDAL)
Escuela Superior de Ingeniería

Máster Propio en Inteligencia Artificial Avanzada y Aplicada (IA3)

Comparativa de arquitecturas post-transformers para la traducción automática de texto

Trabajo fin de estudio presentado por:	Ramírez Herrera, Felipe
Tipo de trabajo:	Comparativa de Soluciones
Director/a:	Soria Olivas, Emilio
Tutor/a:	Manuel Valero Laparra
Fecha:	29 de abril de 2025

Tabla de contenido

Índice de cuadros

Índice de gráficos

Índice de anexos

1 INTRODUCTION

1.1 Motivación

Después de la popularización de los Transformers (Vaswani et al., 2017) , que revolucionaron el procesamiento del lenguaje natural y otros campos de la inteligencia artificial, han surgido varias variantes y avances significativos en la arquitectura que son aplicables al procesamiento del lenguaje natural y en particular a la traducción automática de textos.

Cada una de estas arquitecturas representa un avance significativo en el campo del procesamiento del lenguaje natural y la inteligencia artificial. Comprender sus diferencias, fortalezas y limitaciones permite a los investigadores y estudiantes mantenerse actualizados con el estado del arte y los últimos avances tecnológicos en NLP (Procesamiento del Lenguaje Natural).

Estos avances reflejan una tendencia hacia la mejora de la eficiencia computacional, la capacidad de manejar secuencias más largas y la adaptabilidad a diferentes tipos de datos y tareas dentro del campo del procesamiento del lenguaje natural y más allá.

Sin embargo, la eficiencia computacional es crucial en la implementación práctica de modelos de inteligencia artificial. Al comparar estas arquitecturas, se pueden evaluar aspectos como el tiempo de entrenamiento, la complejidad computacional y los requisitos de memoria, lo que ayuda a seleccionar modelos que sean más eficientes en términos de recursos.

Algunas de estas arquitecturas representan variantes o extensiones de modelos existentes (por ejemplo, *Transformer* y MoE). Comprender cómo estas variaciones afectan el rendimiento puede inspirar investigaciones futuras sobre nuevas mejoras o adaptaciones de estas arquitecturas para problemas específicos.

Consecuentemente, realizar una comparativa permite identificar cuál de estas arquitecturas podría ser más efectiva para diferentes contextos de aplicación.

La capacidad de generalización de un modelo es crucial para su aplicación en entornos del mundo real. Al evaluar estas arquitecturas en una variedad de conjuntos de datos y tareas, se puede entender mejor cómo manejan la generalización y la adaptabilidad a diferentes dominios lingüísticos y contextos.

Realizar una comparativa sistemática y rigurosa entre estas arquitecturas no solo proporciona información práctica para aplicaciones actuales, sino que también contribuye al conocimiento científico general sobre la efectividad de diferentes enfoques en el campo del procesamiento del lenguaje natural y la inteligencia artificial.

1.2 Planteamiento del problema

El trabajo final de máster en Inteligencia Artificial se enfoca en realizar una comparativa exhaustiva de diversas arquitecturas de redes neuronales para el modelado de traducción neuronal (NMT) que han surgido después de la introducción de Transformers.

Los *Transformers*, han demostrado ser efectivos en tareas como la traducción automática y el procesamiento del lenguaje natural. En los últimos años, la traducción neuronal automática ha avanzado significativamente con el desarrollo de nuevas arquitecturas más allá de los modelos Transformer "originales" (Vaswani et al., 2017).

Este estudio se centra en evaluar y comparar varias de estas arquitecturas avanzadas para entender sus fortalezas y limitaciones en tareas de traducción neuronal.

1.3 Estructura de la memoria

2 CONTEXTO Y ESTADO DEL ARTE

2.1 Traducción automática

La traducción automática (NMT, *Neural Machine Translation*) es un enfoque avanzado en el campo de la traducción automática que utiliza redes neuronales profundas (*deep learning*) para traducir texto de un idioma a otro (Koehn, 2020).

En NMT, tanto el proceso de traducción como el aprendizaje se realizan utilizando redes neuronales profundas. Estas redes consisten en capas de neuronas interconectadas que procesan información de manera secuencial y aprenden representaciones abstractas de los datos.

A diferencia de los sistemas de traducción automática estadística tradicionales, que se basan en reglas y características lingüísticas específicas, los modelos NMT aprenden directamente de los datos utilizando un enfoque de aprendizaje *end-to-end*. Esto significa que el modelo de traducción se

entrena para convertir directamente secuencias de palabras de un idioma a otro, sin requerir etapas intermedias de extracción de características o alineación de frases.

El objetivo de aprendizaje es encontrar la secuencia de destino correcta dada la secuencia de origen, lo cual puede verse como un problema de clasificación de alta dimensionalidad que intenta mapear las dos oraciones en el espacio semántico. En todos los principales modelos modernos de NMT, este proceso se puede dividir en un paso de codificación y un paso de decodificación, y así separar funcionalmente todo el modelo (Yang et al., 2024). Consecuentemente, un modelo NMT generalmente sigue una arquitectura básica de codificador-decodificador (*encoder-decoder*) que se detalla más adelante.

Inicialmente, el modelo procesa la secuencia de palabras de entrada en un espacio de representación vectorial (*embedding*) de alta dimensionalidad. Este *embedding* captura el significado semántico de la frase en el idioma original.

Para mejorar la calidad de la traducción y manejar eficientemente secuencias largas, muchos modelos NMT utilizan mecanismos de atención. La atención permite al modelo enfocarse en partes relevantes de la entrada durante la fase de decodificación, mejorando así la coherencia y la precisión de la traducción.

Los modelos NMT requieren grandes volúmenes de datos paralelos (pares de oraciones traducidas) para entrenarse efectivamente. Durante el entrenamiento, el modelo ajusta los pesos de las conexiones neuronales para minimizar la diferencia entre las traducciones generadas y las traducciones reales en el conjunto de datos de entrenamiento.

A medida que se introducen más datos y se optimizan las técnicas de entrenamiento y arquitecturas de red, los modelos NMT han demostrado ser capaces de generar traducciones de mayor calidad que los sistemas tradicionales.

En este sentido Eisenstein (2019) destaca la relación entre la hipótesis distribucional y la capacidad de los modelos de *deep learning* para aprender la traducción automática neuronal (NMT) sin necesidad de conocer explícitamente reglas gramaticales o sintácticas se fundamenta en cómo estos modelos procesan y aprenden a partir de grandes cantidades de datos.

La hipótesis distribucional propuesta por Harris (1954) y popularizado por Firth (1957) es un principio lingüístico que sugiere que las palabras que aparecen en contextos similares tienden a tener

significados similares. De forma simplificada, el contexto en el que aparece una palabra proporciona pistas sobre su significado. Esta idea subraya que el significado de una palabra se puede inferir a partir de las palabras que suelen aparecer en su proximidad. Este concepto se fundamenta en la observación de que el contexto de una palabra es crucial para comprender su significado.

De este modo la hipótesis distribucional proporciona el marco teórico que respalda la efectividad de los modelos modernos de traducción automática basados en *deep learning*, para aprender y aplicar conocimientos lingüísticos sin necesidad de reglas gramaticales explícitas.

Así, esta premisa de la hipótesis distribucional se traduce en cómo los modelos de *deep learning* representan y aprenden de las palabras. En esencia, los modelos de lenguaje modernos implementan esta hipótesis al aprender representaciones vectoriales de palabras que reflejan sus contextos. Estas representaciones vectoriales implementan el aprendizaje de representaciones al generar *embeddings* de palabras. Como se mencionó previamente, los *embeddings* son vectores que capturan el significado contextual de las palabras, alineándose con la hipótesis distribucional al representar palabras que comparten contextos similares con vectores similares.

Este principio es esencial para los modelos de *deep learning*, ya que estos aprenden patrones estadísticos a partir de datos (Bishop & Bishop, 2024). En el caso de la traducción automática neuronal, el modelo NMT no necesita conocimientos explícitos sobre las reglas gramaticales o sintácticas de los idiomas involucrados. En cambio, aprende a asociar patrones de palabras y frases enteras en un idioma con sus correspondientes en otro idioma.

Los modelos de *deep learning*, como las redes neuronales profundas utilizadas en NMT, son capaces de aprender representaciones distribuidas de palabras y frases (Eisenstein, 2019). Esto significa que cada palabra o frase se representa mediante vectores numéricos en un espacio de alta dimensionalidad.

Durante el entrenamiento, el modelo NMT ajusta estos vectores (*embedding*) y los pesos de las conexiones neuronales para minimizar el error de traducción en un conjunto de datos paralelos (pares de frases en diferentes idiomas).

A medida que el modelo procesa más ejemplos y ajusta sus parámetros, captura de manera implícita reglas sintácticas y gramaticales. Por ejemplo, aprende a generar traducciones

gramaticalmente correctas al aprender qué estructuras de frases son más comunes y efectivas en cada idioma.

La capacidad de los modelos de *deep learning* para aprender sin necesidad de reglas explícitas radica en su habilidad para generalizar a partir de datos (Prince, 2023). En lugar de depender de reglas predefinidas, estos modelos aprenden a través de ejemplos y exposición a datos diversos mediante *transducción* (Gammerman et al., 2013).

Esto significa que el modelo NMT puede capturar fenómenos lingüísticos complejos que podrían no estar explícitamente codificados en reglas gramaticales tradicionales. Por ejemplo, puede aprender a manejar variaciones idiomáticas, ambigüedades contextuales y estructuras sintácticas específicas de cada idioma (Koehn, 2020).

Los **modelos de lenguaje** como los Transformers (por ejemplo, BERT, GPT), se basan en el aprendizaje de representaciones para comprender y generar texto. Estos modelos utilizan grandes cantidades de datos y aprenden a predecir palabras o frases en función de su contexto. La efectividad de estos modelos se basa en gran medida en la capacidad de capturar y utilizar el contexto para entender el significado de las palabras (Kamath et al., 2022).

El **aprendizaje de representaciones** es un enfoque en el aprendizaje automático donde se busca aprender representaciones significativas y útiles de los datos, a menudo a partir de datos sin etiquetar. En el contexto del procesamiento del lenguaje natural (NLP), esto implica aprender representaciones vectoriales de palabras que capturen sus significados en función de los contextos en los que aparecen (Liu et al., 2023).

A medida que los modelos de lenguaje se entrenan en grandes cuerpos de texto, mejoran su capacidad para capturar relaciones contextuales sutiles y complejas entre palabras. Este proceso de aprendizaje continuo mejora la efectividad de los modelos para tareas de NLP, como la traducción automática, el análisis de sentimientos o la generación de texto, al afinar las representaciones sustentadas por la hipótesis distribucional.

Estos modelos aprenden a través de patrones estadísticos derivados de grandes volúmenes de datos, lo que les permite generar traducciones precisas y naturales entre idiomas.

2.2 Transducción

Muchas tareas de aprendizaje automático pueden expresarse como la transformación, o transducción, de secuencias de entrada en secuencias de salida: reconocimiento de voz, traducción automática, predicción de la estructura secundaria de proteínas y síntesis de voz, por mencionar algunas (Graves, 2012).

La capacidad de transformar y manipular secuencias es una parte crucial de la inteligencia humana: todo lo que se sabe sobre el mundo llega en forma de secuencias sensoriales, y todo lo que se hace para interactuar con el mundo requiere secuencias de acciones y pensamientos. La creación de transductores automáticos de secuencias parece ser, por lo tanto, un paso importante hacia la inteligencia artificial. Un problema importante al que se enfrentan estos sistemas es cómo representar la información secuencial de manera que sea invariante, o al menos robusta, frente a distorsiones secuenciales. Además, esta robustez debe aplicarse tanto a las secuencias de entrada como a las de salida.

La transducción de secuencias, también conocida como modelado de secuencia a secuencia, es una tarea de aprendizaje automático que implica convertir una secuencia de entrada en una secuencia de salida, potencialmente de diferentes longitudes.

El término se utiliza en algunas aplicaciones de redes neuronales recurrentes en problemas de predicción de secuencias, como algunos problemas en el dominio del procesamiento de lenguaje natural.

La transducción (o aprendizaje transductivo) se usa en el campo de la teoría del aprendizaje estadístico para referirse a la predicción de ejemplos específicos dados ejemplos específicos de un dominio. Vale la pena distinguir el proceso "inductivo" consiste en derivar la función a partir de los datos dados de la "deducción" que implica derivar los valores de la función dada para puntos de interés y, finalmente, "transducción" es derivar los valores de la función desconocida para puntos de interés a partir de los datos dados.

El modelo de estimación del valor de una función en un punto de interés describe un nuevo concepto de inferencia: pasar de lo particular a lo particular. A este tipo de inferencia se le denomina *inferencia transductiva*. Es importante destacar que este concepto de inferencia aparece cuando se desea obtener el mejor resultado a partir de una cantidad restringida de información.

Clásicamente, la transducción se ha utilizado al hablar de lenguaje natural, como en el campo de la lingüística. Por ejemplo, existe la noción de "*gramática de transducción*" que se refiere a un conjunto de reglas para transformar ejemplos de un idioma a otro.

Una gramática de transducción describe un par de lenguajes estructuralmente correlacionados y genera pares de oraciones, en lugar de oraciones individuales. La oración en el idioma origen es (se supone que es) una traducción de la oración en el idioma destino.

Este uso de la transducción al hablar de teoría y traducción automática clásica influye en el uso del término cuando se trata de la predicción moderna de secuencias con redes neuronales recurrentes en tareas de procesamiento de lenguaje natural.

Más generalmente, la transducción se usa en tareas de predicción de secuencias en NLP, específicamente en traducción. Por ejemplo, Grefenstette et al. (2015) describen la transducción como la asignación de una cadena de entrada a una cadena de salida.

Muchas tareas de procesamiento de lenguaje natural (NLP) pueden verse como problemas de transducción, es decir, aprender a convertir una cadena en otra. La traducción automática es un ejemplo prototípico de transducción y los resultados recientes indican que las RNN tienen la capacidad de codificar largas cadenas de entrada y producir traducciones coherentes.

Un transductor se define de manera estricta como un modelo que produce una salida por cada paso de tiempo de entrada proporcionado. Esto se relaciona con el uso lingüístico, específicamente con los transductores de estados finitos. Sin, embargo un Transductor Neuronal (Jaitly et al., 2015) es una clase más general de modelos de aprendizaje de secuencia a secuencia y puede producir bloques de salidas (posiblemente de longitud cero) a medida que llegan bloques de entradas, cumpliendo así con la condición de ser “online”. El modelo genera salidas para cada bloque utilizando una RNN transductora que implementa un modelo de secuencia a secuencia (Seq-to-Seq).

2.3 Arquitecturas y modelos

De acuerdo con Vardasbi et al. (2023) la traducción automática se modela usando una red neuronal. En los modelos de codificador-decodificador, como el Transformer, el modelo tiene dos componentes principales: un codificador para capturar las dependencias del lado de la fuente y un decodificador para capturar las dependencias del lado del objetivo y las dependencias entre fuente y objetivo. Alternativamente, la traducción automática (MT) puede tratarse como una tarea de

Modelado del Lenguaje, donde el modelo (solo decodificador) se entrena con las oraciones fuente y objetivo concatenadas, separadas por un token especial [SEP].

Un modelo de secuencia a secuencia (Seq-to-Seq) típicamente presenta una arquitectura compuesta por uno o más codificadores y uno o más decodificadores.

De acuerdo con Anastasopoulos & Chiang (2018) y Sperber et al. (2020) definir la arquitectura con dos o más decodificadores implicaría "dual-tasking" (Le et al., 2020) y (Yu et al., 2022), mientras que definirla con dos codificadores o "cruzar" los decodificadores conllevaría "multi-modality" a nivel de entrada o salida (Sun et al., 2023).

Un modelo codificador-decodificador mapea una secuencia de entrada a una secuencia objetivo con ambas secuencias de longitud arbitraria (Sutskever et al., 2014). Este tipo de modelo tienen aplicaciones que van desde la traducción automática hasta la predicción de series temporales. Más específicamente, este mecanismo utiliza una RNN, o cualquiera de sus variantes como *LSTM* (*Long Short Term Memory*) o *GRU* (*Gated Recurrent Unit*), para mapear la secuencia de entrada a un vector de longitud fija, y otra RNN o una de sus variantes para decodificar la secuencia objetivo a partir de ese vector.

Tanto el "*encoder"* y el *"decoder"* son componentes clave en modelos de redes neuronales diseñados para tareas de procesamiento de lenguaje natural, trabajando juntos para transformar y generar secuencias de texto de manera efectiva y precisa.

El *"encoder"*, o codificador es la parte de la red que transforma la entrada (como texto o una secuencia de palabras) en una representación numérica o vectorial. Esta representación captura el significado semántico y estructural de la entrada original en un formato que la red neuronal puede entender y procesar.

Por su parte, el *decoder*, o decodificador, es la parte de la red neuronal que utiliza la representación numérica generada por el *encoder* para producir una salida en la forma deseada. En el contexto de traducción automática, el *decoder* toma la representación interna de la frase en el idioma origen, generada por el *encoder*, y la transforma en la secuencia de palabras en el idioma destino.

El proceso de *encoder-decoder* es fundamental en tareas de procesamiento de lenguaje natural como la traducción automática y la generación de texto, donde el *encoder* ayuda a comprender

y representar el significado de la entrada, mientras que el *decoder* utiliza esta representación para producir la salida deseada en un formato comprensible para los humanos.

El estudio de S. Wang et al. (2021) concluye que los modelos de lenguaje grandes tienen un gran potencial para mejorar la traducción automática sin necesidad de arquitectura de codificador-decodificador, demostrando su capacidad para tareas que involucran tanto comprensión como generación de lenguaje

2.4 Atención

El trabajo de Bahdanau et al. (2014) fue fundamental para el desarrollo de modelos de traducción automática basados en redes neuronales, particularmente en la evolución de los modelos *Encoder-Decoder* con mecanismos de atención, este aporte es relevante particularmente por:

- Introducción del mecanismo de atención que permite que el modelo se enfoque en partes específicas de la secuencia de entrada durante la traducción, mejorando la precisión y la fluidez de las traducciones generadas.
- El enfoque propuesto aborda de manera integral dos problemas clave en la traducción automática: la alineación precisa de las palabras en las dos lenguas y la generación de la traducción. Al aprender simultáneamente a alinear y traducir, el modelo puede mejorar la coherencia y la fidelidad de las traducciones.
- Mejora en la fluidez y la coherencia: Al integrar el aprendizaje de la alineación junto con el proceso de traducción, el modelo puede capturar mejor las correspondencias entre palabras y frases en diferentes idiomas. Esto resulta en traducciones más precisas y naturales, reduciendo errores de alineación y mejorando la cohesión del texto traducido.

Sin duda el aporte de Bahdanau et al. (2014) ha inspirado numerosos estudios y avances en la arquitectura de modelos de traducción automática basados en redes neuronales, promoviendo el uso generalizado de mecanismos de atención en diversos modelos y aplicaciones. El éxito del enfoque propuesto en este artículo allanó el camino para el desarrollo de modelos más sofisticados y efectivos en traducción automática, como los Transformers. Estos modelos han revolucionado el campo al mejorar aún más la capacidad de manejar contextos largos y capturar relaciones complejas entre las palabras.

Al distinguir entre secuencias "en distribución"[1] y "fuera de distribución"[2], se puede evaluar cómo un modelo generaliza a ejemplos que no están representados en el conjunto de entrenamiento (Z. Wang, 2023). Esto ayuda a determinar si el modelo ha aprendido una función subyacente general que puede aplicarse a nuevas secuencias de diferentes longitudes (fuera de distribución), o si solo está aproximando los datos que vio durante el entrenamiento (en distribución).

De esta forma si un modelo no puede generalizar bien a secuencias de longitud diferente a las de entrenamiento, entonces no ha aprendido la función subyacente real y solo ha aprendido a ajustar los datos en distribución.

Por su parte Z. Wang (2023) destaca que la atención mejora enormemente la eficiencia del aprendizaje, al reducir tanto la complejidad del modelo como la complejidad de la muestra, así como la robustez del aprendizaje, en términos de rendimiento de generalización y problemas de sobreajuste. Sin embargo, la atención no supera la limitación de generalización fuera de distribución, ya que no cambia la naturaleza autorregresiva de los modelos. No obstante, la impresionante eficiencia de aprendizaje acelerada por la atención refleja su motivación original, es decir, "aprender a alinear" (Bahdanau et al., 2015).

Si bien Z. Wang (2023) señala que los modelos de red neuronal recurrente de secuencia a secuencia (RNN seq2seq) se han utilizado para aprender cuatro tareas de transducción: identidad, inversión, reducción total y copia cuadrática. Estas tareas de transducción se han estudiado tradicionalmente usando transductores de estado finito y se les ha atribuido una creciente complejidad. El autor señala que se ha observado que los modelos RNN seq2seq solo pueden aproximar un mapeo que se ajusta a los datos de entrenamiento o datos en distribución, en lugar de aprender las funciones subyacentes. Aunque el mecanismo de atención mejora la eficiencia y robustez del aprendizaje, no supera la limitación de generalización para datos “fuera de distribución”.

De acuerdo con Z. Wang (2023), los modelos con atención (*attentional*) muestran que la atención mejora significativamente la capacidad de los modelos para ajustarse a los datos de

[1] Se refiere a secuencias de entrada cuya longitud está dentro del rango de longitudes de secuencias usadas durante el entrenamiento del modelo. Por ejemplo, si un modelo fue entrenado con secuencias de longitud entre 6 y 15, entonces una secuencia de longitud 10 sería "en distribución".

[2] Se refiere a secuencias de entrada cuya longitud no está dentro del rango de longitudes usadas durante el entrenamiento. Siguiendo el mismo ejemplo, una secuencia de longitud 5 o 16 sería "fuera de distribución".

entrenamiento y generalizar a los datos de prueba. Los modelos con atención siempre logran una mejor precisión agregada en secuencias completas tanto en los conjuntos de entrenamiento como en los de prueba, y tienen una menor variabilidad entre el entrenamiento y la prueba en comparación con los modelos sin atención (*attention-less*). Además, los modelos con atención superan a los modelos sin atención en la generalización a ejemplos fuera de distribución. En otras palabras, los modelos *RNN seq2seq* con atención son más fuertes en el aprendizaje dentro de la distribución y tienen una capacidad de generalización fuera de distribución relativamente mejor en comparación con los modelos sin atención.

Aunque la atención mejora la capacidad del modelo para manejar datos fuera de distribución en comparación con los modelos sin atención, el problema de generalización aún persiste, aunque en menor grado.

2.5 Transformers

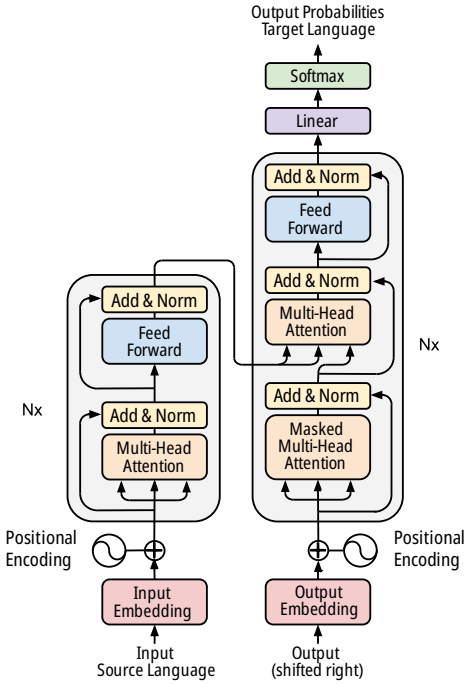
Los *Transformers* (Vaswani et al., 2017) representan una alternativa innovadora al enfoque tradicional de las redes neuronales recurrentes (RNN) en el campo del procesamiento de lenguaje natural y otras tareas secuenciales.

El *Transformer* es un modelo prominente en el aprendizaje profundo ampliamente adoptado en campos como el procesamiento del lenguaje natural (NLP), visión por computadora (CV) y procesamiento de voz. Inicialmente propuesto como un modelo secuencia a secuencia para la traducción automática, los modelos pre-entrenados basados en *Transformer* han logrado el estado del arte en diversas tareas, consolidándolo como la arquitectura principal en NLP, especialmente para PTMs.

Además de aplicaciones en lenguaje, el *Transformer* ha sido utilizado en CV, procesamiento de audio e incluso en disciplinas como química y ciencias de la vida. Diversas variantes del *Transformer*, conocidas como *X-formers*, han sido propuestas para mejorar su eficiencia en el procesamiento de secuencias largas, su capacidad de generalización y adaptación a tareas específicas (T. Lin et al., 2022).

El *Transformer* básico es un modelo secuencia a secuencia que consta de un codificador y un decodificador, cada uno compuesto por una pila de L bloques idénticos. Cada bloque del codificador se compone principalmente de un módulo de autoatención multi-cabeza y una red de alimentación hacia adelante (FFN) por posición. Para construir un modelo más profundo, se utiliza una conexión

residual alrededor de cada módulo, seguida de un módulo de Normalización de Capa. Comparados con los bloques del codificador, los bloques del decodificador insertan adicionalmente módulos de atención cruzada entre los módulos de autoatención multi-cabeza y las FFNs por posición. Además, los módulos de autoatención en el decodificador están adaptados para evitar que cada posición atienda a posiciones subsecuentes.



Al comparar los Transformers frente a RNN tradicionales, se pueden destacar las siguientes innovaciones:

- Arquitectura sin recurrencia: A diferencia de las RNN tradicionales, que procesan secuencias de manera secuencial y recurrente, los Transformers no tienen conexiones recurrentes. En su lugar, utilizan mecanismos de atención para capturar las relaciones entre las diferentes partes de la secuencia de entrada simultáneamente.
- Mecanismo de atención: Los Transformers utilizan capas de atención que calculan pesos para cada par de elementos en la secuencia de entrada, permitiendo que el modelo "atienda" partes específicas de la secuencia que son relevantes para la tarea en cuestión. Esto facilita la captura de dependencias a largo plazo y mejora la capacidad del modelo para manejar secuencias de longitud variable (Chaudhari et al., 2021) y (Brauwers & Frasincar, 2023).

- Paralelización: Debido a su naturaleza no recurrente y al uso eficiente de la atención, los Transformers son más fácilmente paralelizables en comparación con las RNN. Esto los hace más rápidos de entrenar y de utilizar en aplicaciones prácticas.

Los autores T. Lin et al. (2022) destacan a la autoatención (self-attention) como la pieza central de los Transformers y mecanismo flexible para manejar entradas de longitud variable. Se puede entender como una capa completamente conectada en la que los pesos se generan dinámicamente a partir de las relaciones entre pares de entradas (*pairwise*).

Entre las ventajas de la autoatención se pueden mencionar:

1. **Longitud del camino máxima**: La autoatención tiene la misma longitud máxima del camino que las capas completamente conectadas, lo que la hace adecuada para modelar dependencias a largo alcance. En comparación con las capas completamente conectadas, la autoatención es más eficiente en términos de parámetros y más flexible para manejar entradas de longitud variable.
2. **Campo receptivo y profundidad de red**: Debido al campo receptivo limitado de las capas convolucionales, generalmente se necesita apilar una red profunda para tener un campo receptivo global. Por otro lado, la longitud máxima del camino constante permite que la autoatención modele dependencias a largo alcance con un número constante de capas.
3. **Operaciones paralelizadas**: Las operaciones secuenciales constantes y la longitud máxima del camino hacen que la autoatención sea más paralelizable y más eficiente en el modelado de dependencias a largo alcance en comparación con las capas recurrentes.

Del mismo modo, al considerar el sesgo inductivo o suposiciones que un modelo hace sobre la distribución de los datos para simplificar el aprendizaje (Bishop & Bishop, 2024). De esta forma T. Lin et al. (2022) destaca que cada tipo de red neuronal tiene diferentes sesgos inductivos que afectan cómo procesan la información.

En el caso de las redes CNN (Convolutional Neural Networks) imponen sesgos inductivos de *invariancia de traducción* y *localidad*. Esto significa que las CNNs son buenas para reconocer patrones que aparecen en diferentes posiciones (traducción) y para enfocarse en áreas locales de la imagen a través de funciones de *kernel* compartidas.

Por su parte los enfoques previos de recurrencia como LSTM o GRU tienen sesgos inductivos de *invariancia temporal* y *localidad* gracias a su estructura Markoviana. Esto les permite manejar datos secuenciales, como el texto o las series temporales, tratando cada paso de la secuencia como dependiente del anterior de manera local.

Por el contrario**,** la arquitectura de *Transformer* tiene muy pocos supuestos sobre la estructura de los datos. Esto significa que no impone sesgos inductivos específicos como las CNNs o las RNNs, lo que la convierte en una arquitectura universal y flexible. Sin embargo, esta flexibilidad también puede llevar a que el *Transformer* se sobreajuste cuando se entrena con datos de pequeña escala, ya que la falta de sesgos estructurales hace que sea más propenso a captar ruido en lugar de patrones significativos (T. Lin et al., 2022).

A pesar de su arquitectura innovadora, los Transformers mantienen el marco general de los modelos *Encoder*/*Decoder* al separar claramente las fases de codificación y decodificación. Esto los hace altamente efectivos para tareas como la traducción automática, donde la secuencia de entrada (por ejemplo, una oración en un idioma) se codifica en una representación semántica universal, que luego se decodifica en la secuencia de salida (la misma oración traducida a otro idioma).

Como lo menciona T. Lin et al. (2022), la arquitectura *Transformer* puede ser utilizada de tres formas diferentes:

- Codificador-Decodificador. Se utiliza la arquitectura completa del *Transformer* como se introdujo en anteriormente. Esto se utiliza típicamente en modelado secuencia a secuencia (por ejemplo, traducción automática neuronal).
- Solo codificador. Solo se utiliza el codificador y las salidas del codificador se utilizan como representación de la secuencia de entrada. Esto se usa generalmente para problemas de clasificación o etiquetado de secuencias.
- Solo decodificador. Solo se utiliza el decodificador, donde también se elimina el módulo de atención cruzada codificador-decodificador. Esto se utiliza típicamente para generación de secuencias, como modelado de lenguaje.

Siguiendo esta misma línea, Xiao & Zhu (2023) profundizan en variantes del *Transformer* que se utilizan en diversas aplicaciones, como modelos *encoder-only* (usados en clasificación de texto), *decoder-only* (para generación de texto), y el modelo *encoder-decoder* (para tareas como la traducción).

Además, se abordan mejoras recientes, como cambios en los mecanismos de atención y nuevas técnicas para codificar información posicional, algunas de las cuales se abordan a continuación.

Si bien los Transformers representan una evolución significativa respecto a las RNN tradicionales al introducir mecanismos de atención y mejorar la capacidad de manejar dependencias a largo plazo. A pesar de estas diferencias, siguen siendo modelos *Encoder* / *Decoder*, destacando por su eficiencia y rendimiento en una amplia gama de aplicaciones de procesamiento de lenguaje natural y más allá.

En el modo codificador-decodificador, generalmente hay múltiples módulos de auto-atención multi-cabezal, incluyendo una auto-atención estándar tanto en el codificador como en el decodificador, junto con una atención cruzada codificador-decodificador que permite al decodificador utilizar información del codificador. Esto influye en el diseño del mecanismo de auto-atención. En el modo codificador, no hay restricción o condición de que el mecanismo de auto-atención deba ser causal, es decir, dependiente únicamente de los tokens presentes y pasados. En el entorno codificador-decodificador, la auto-atención utilizada en el decodificador (es decir, a través de posiciones de decodificación) debe ser causal, ya que cada paso de decodificación auto-regresivo solo puede depender de tokens anteriores, mientras que la auto-atención utilizada en el codificador no necesita serlo. Cumplir con este requisito puede resultar desafiante para muchos diseños eficientes de auto-atención. (Tay et al., 2022).

Los costos computacionales de los Transformers son influenciados por varios factores clave. Primero, la complejidad computacional surge principalmente del mecanismo de atención. Específicamente, calcular la matriz de atención implica un costo cuadrático en relación con la longitud de la secuencia de entrada N, denotado como $N \times N$. Esta complejidad cuadrática es evidente en operaciones como la multiplicación de matrices para las consultas Q y las llaves K, que por sí solas consumen tiempo y memoria proporcional a N^2. Esto limita la utilidad de los modelos auto-atentivos en aplicaciones que requieren el procesamiento de secuencias largas, afectando especialmente durante el entrenamiento debido a las actualizaciones de gradiente, aunque menos durante la inferencia donde los gradientes no se calculan (Tay et al., 2022).

El costo cuadrático de la auto-atención afecta tanto la velocidad de las fases de entrenamiento como de inferencia. Sin embargo, el costo computacional total de un *Transformer* también incluye contribuciones significativas de las capas de alimentación hacia adelante presentes en cada bloque

del *Transformer*. Estas capas, aunque lineales en complejidad con respecto a la longitud de la secuencia, aún representan una parte no trivial de la carga computacional. Se han realizado esfuerzos para mitigar estos costos, como explorar la dispersión en las redes de alimentación hacia adelante para escalarlas sin un aumento correspondiente en los costos computacionales.

La eficiencia en los mecanismos de atención y en la computación general del modelo son consideraciones algo independientes pero entrelazadas. Algunos métodos de atención eficientes tienen como objetivo reducir explícitamente las longitudes de secuencia, ahorrando así costos de computación tanto en las capas de atención como en las de alimentación hacia adelante. Gestionar estos compromisos entre eficiencia y demandas computacionales es una tarea compleja, abordada frecuentemente en investigaciones recientes que exploran diversas estrategias de optimización y arquitecturas de modelos.

Como mencionan Katharopoulos et al. (2020), se han realizado pocos esfuerzos para comprender la autoatención desde una perspectiva teórica. El mismo autor menciona que se han propuesto formulaciones basadas en *kernels*, en la que la atención se interpreta como la aplicación de un suavizador de *kernels* sobre las entradas, con las puntuaciones de los *kernels* representando la similitud entre las entradas. Esta formulación facilita la comprensión de los componentes de la atención y la integración de las incrustaciones posicionales. En contraste, la formulación del *kernel* se utiliza para acelerar el cálculo de la autoatención y reducir su complejidad computacional. Además, se observa que, si se aplica un *kernel* con puntuaciones de similitud positivas a las consultas y claves, la atención lineal converge normalmente. Más recientemente, demostraron teórica y empíricamente que una autoatención multi-cabeza con un número suficiente de cabezas puede replicar cualquier capa convolucional. En lugar de esto, se muestra que una capa de autoatención entrenada con un objetivo autorregresivo puede ser interpretada como una red neuronal recurrente, lo que permite acelerar significativamente el tiempo de inferencia en modelos de transformadores autorregresivos.

Como se ha mencionado, la autoatención juega un papel crucial en la arquitectura *Transformer*, pero presenta dos desafíos importantes en aplicaciones prácticas (T. Lin et al., 2022):

1. **Complejidad:** La complejidad de la autoatención es $O(T^2 \cdot D)$, donde T es la longitud de la secuencia y D es la dimensión del vector de características. Esto significa que el módulo de atención puede volverse un cuello de botella cuando se trata de secuencias largas, ya que el tiempo de cálculo crece cuadráticamente con la longitud de la secuencia.

2. **Prior Estructural:** La autoatención no hace suposiciones sobre la estructura de los datos de entrada, ni siquiera sobre el orden de los elementos en la secuencia. Por lo tanto, la información sobre el orden debe ser aprendida a partir de los datos de entrenamiento. Esto hace que los Transformers, sin pre-entrenamiento, sean propensos al sobreajuste cuando se utilizan con datos pequeños o de tamaño moderado.

Como lo menciona T. Lin et al. (2022), para abordar estos desafíos, se han desarrollado varias mejoras en el mecanismo de atención:

1. **Atención Dispersa (Sparse Attention):** Esta línea de investigación introduce sesgos de escasez en el mecanismo de atención, lo que reduce la complejidad al limitar el número de interacciones necesarias.
2. **Atención Linealizada (Linearized Attention):** En este enfoque, la matriz de atención se descompone en mapas de características de núcleo (kernel). Luego, la atención se calcula en un orden inverso para lograr una complejidad lineal en lugar de cuadrática (Katharopoulos et al., 2020).
3. **Compresión de Prototipos y Memoria (Prototype and Memory Compression):** Estos métodos reducen el número de pares de consultas o memorias clave-valor para disminuir el tamaño de la matriz de atención.
4. **Autoatención de Bajo Rango (Low-rank Self-Attention):** Esta línea de trabajo captura la propiedad de bajo rango de la autoatención (Bhojanapalli et al., 2020), lo que ayuda a reducir la complejidad computacional (Zhang et al., 2022).
5. **Atención con Priori (Attention with Prior):** Esta investigación explora cómo complementar o sustituir la atención estándar con distribuciones de atención previas, lo que puede mejorar el rendimiento al incorporar conocimientos adicionales.
6. **Mecanismo Multi-Cabeza Mejorado (Improved Multi-Head Mechanism):** Este enfoque investiga diferentes mecanismos alternativos para la atención multi-cabeza, buscando mejorar la eficiencia y efectividad del proceso de atención.

Cada una de estas mejoras busca abordar los desafíos asociados con la autoatención, haciéndola más eficiente y efectiva para diferentes aplicaciones y tamaños de datos.

2.6 Universal Transformers

Sobre la base de la arquitectura de *Transformers*, Dehghani et al. (2019) introducen la variante de "*Universal Transformers*" que representa un avance notable en el campo de las redes neuronales y el procesamiento de lenguaje natural. Introduce un enfoque innovador que mejora los modelos *Transformers* estándar al permitir una adaptabilidad dinámica en cada capa del modelo.

A diferencia de los *Transformers* tradicionales, donde todas las capas y posiciones de la secuencia realizan la misma cantidad de operaciones, los "*Universal Transformers*" pueden ajustar dinámicamente la cantidad de pasos de computación en función de las necesidades específicas de cada posición en la secuencia.

Esta capacidad dinámica no solo mejora la eficiencia computacional del modelo al reducir el número de operaciones innecesarias, sino que también potencia su capacidad para capturar dependencias a largo plazo de manera más efectiva.

Los experimentos realizados por Dehghani et al. (2019) demostraron que los *Universal Transformers* superan a los modelos estándar en diversas tareas complejas de procesamiento de lenguaje natural, como la traducción automática y el análisis de sentimientos. Esta superioridad se debe a su capacidad para adaptarse de manera flexible a diferentes partes de la secuencia, capturando relaciones más complejas entre palabras y frases.

2.7 Reformer

El trabajo de Kitaev et al. (2020) es una contribución significativa en la evolución de los modelos *Transformers*, centrada en mejorar la eficiencia computacional y el manejo de secuencias largas. Uno de los aspectos clave de este avance es el uso de técnicas de *hashing*, específicamente el LSH (*Locality Sensitive Hashing*), aplicado a la capa de atención.

El *Reformer* aborda el desafío de la escalabilidad de los modelos Transformers al introducir técnicas que reducen drásticamente el costo computacional y de memoria, permitiendo el procesamiento eficiente de secuencias largas. Esto se logra mediante el uso de aproximaciones eficientes en el cálculo de atención, como *hashing* y atención por *chunking*, que reducen la complejidad cuadrática típica de la atención.

A diferencia de otros modelos que enfrentan problemas con secuencias extensas debido a la memoria requerida, el *Reformer* utiliza técnicas como la atención reversible y la compresión de parámetros para manejar de manera efectiva secuencias de longitud considerable, sin comprometer la calidad de las representaciones aprendidas.

Los experimentos realizados muestran que el *Reformer* mantiene un rendimiento competitivo en comparación con los *Transformers* estándar, pero con una significativa reducción en los requisitos de recursos computacionales. Esto lo hace adecuado para aplicaciones en escenarios donde el procesamiento de grandes volúmenes de datos es crucial, como en la traducción automática y el análisis de texto a gran escala.

En los modelos *Transformer* estándar, el cálculo de atención entre todas las posiciones de una secuencia implica una complejidad computacional cuadrática con respecto a la longitud de la secuencia. Esto puede volverse prohibitivo con secuencias muy largas debido a la gran cantidad de cálculos necesarios.

El *Reformer* aborda este problema mediante el uso de LSH, una técnica que aproxima el cálculo de atención al dividir el espacio de las posiciones en cubetas hash. En lugar de calcular la atención exacta entre todas las posiciones, LSH agrupa posiciones similares en el espacio de representación, reduciendo así la cantidad de pares de atención que deben considerarse explícitamente.

En la práctica, el *Reformer* utiliza LSH para dividir la matriz de consulta y la matriz de clave en segmentos, que luego se atienden de manera aproximada utilizando operaciones de hash. Esto permite que el modelo mantenga un rendimiento competitivo mientras reduce significativamente la carga computacional y la memoria necesaria para procesar secuencias largas.

El *Reformer* ha influido en la investigación al mostrar nuevas vías para mejorar la eficiencia y escalabilidad de los modelos *Transformer*, inspirando desarrollos posteriores en la optimización de atención y la gestión de recursos en redes neuronales de gran escala. Esta técnica innovadora demuestra cómo la investigación continua en optimización puede mejorar radicalmente la eficiencia de las redes neuronales en el procesamiento de lenguaje natural y otras tareas de aprendizaje automático.

2.8 Evolved Transformer

El So et al. (2019) proponen "*The Evolved Transformer*" como un enfoque novedoso para mejorar el rendimiento de los modelos Transformer mediante una búsqueda evolutiva automatizada de arquitecturas neuronales.

La propuesta de un *Evolved Transformer* explora cómo optimizar el diseño estructural del *transformer* a través de una técnica de búsqueda evolutiva, mejorando los modelos estándar a nivel de arquitectura.

Para ello se propone un algoritmo evolutivo que optimiza arquitecturas de Transformer, utilizando un método llamado "Búsqueda de Arquitectura con Hurdles Dinámicos Progresivos" (PDH, por sus siglas en inglés). Este algoritmo ajusta la arquitectura del modelo a medida que evoluciona, creando barreras (hurdles) que los modelos deben superar en términos de rendimiento en pruebas de validación. Si un modelo no supera una de estas barreras, se detiene su entrenamiento, evitando así la computación innecesaria. Los modelos que pasan todas las pruebas reciben un entrenamiento adicional y continúan mejorando.

El enfoque se basa en seleccionar entre múltiples configuraciones arquitectónicas, que incluyen variaciones en las capas, funciones de activación y métodos de normalización. Cada "modelo hijo" tiene una representación genética que describe sus componentes, y los mejores modelos se seleccionan para continuar evolucionando. Este proceso garantiza que se exploren múltiples combinaciones, lo que lleva a encontrar arquitecturas más eficientes.

De acuerdo al documento original (So et al., 2019), esta arquitectura logró superar a la arquitectura Transformer estándar en varias tareas, incluyendo la traducción automática. Los autores realizaron estudios de ablación para entender qué mutaciones específicas en la arquitectura del Transformer contribuyeron a esta mejora en el rendimiento. Entre los hallazgos clave, se descubrió que las capas de atención al codificador eran esenciales para el buen desempeño, y que la eliminación de la normalización de capas degradaba significativamente la calidad del modelo.

Este método automatizado permite explorar eficientemente grandes espacios de diseño arquitectónico sin intervención humana directa, reduciendo el tiempo y esfuerzo necesario para encontrar configuraciones óptimas. La búsqueda evolutiva con PDH es particularmente útil porque ajusta dinámicamente los pasos de entrenamiento y los recursos según el rendimiento de los modelos

en desarrollo, lo que ayuda a evitar el sobreentrenamiento o el uso excesivo de recursos computacionales en modelos que no muestran promesas.

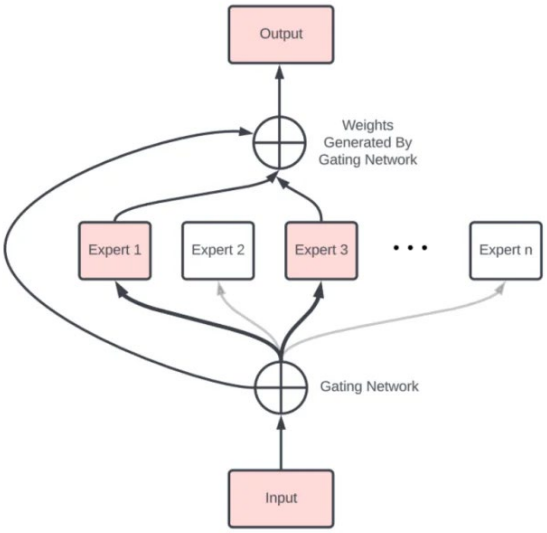
2.9 Mixture of Experts (MoE)

El uso de Modelos de Expertos Dispersos (MoE) en la traducción automática ofrece varias ventajas significativas:

En los MoE, solo un subconjunto de expertos se activa para cada entrada. Esto reduce la cantidad de cálculos necesarios durante la inferencia y el entrenamiento, ya que no todos los expertos contribuyen a cada paso. Esta activación selectiva permite que el modelo maneje grandes cantidades de datos de entrada sin un aumento proporcional en el costo computacional.

Al activar solo una parte de los expertos, se reduce el consumo de memoria y la carga computacional. Esto es especialmente valioso cuando se trabajan con grandes modelos de traducción automática que pueden ser muy costosos en términos de recursos.

Los expertos en un MoE pueden especializarse en diferentes aspectos del proceso de traducción, como distintos pares de idiomas, contextos específicos, o tipos de texto (técnico, literario, etc.). Esta especialización puede mejorar la calidad de la traducción al permitir que el modelo maneje de manera más efectiva las variaciones en el lenguaje y el contenido.



El mecanismo de *gating* en MoE permite que el modelo adapte dinámicamente qué expertos se activan según la entrada específica. Esto puede ayudar a mejorar la precisión y la relevancia de las traducciones al seleccionar los expertos más adecuados para cada entrada.

MoE permite construir modelos más grandes y complejos sin un aumento lineal en el costo computacional. Esto significa que se pueden usar modelos con más expertos sin tener que aumentar significativamente el poder de procesamiento necesario, lo que facilita el escalado para tareas de traducción más complejas.

La capacidad de manejar eficientemente grandes volúmenes de datos y secuencias largas es crucial para la traducción automática, especialmente cuando se trabaja con textos extensos o complejos. Los modelos MoE puede ayudar a gestionar estos desafíos sin una penalización significativa en términos de tiempo y recursos.

La capacidad de añadir o ajustar expertos en un modelo MoE permite una adaptación más fácil a nuevos idiomas o dominios. Por ejemplo, se pueden añadir expertos especializados en un nuevo par de idiomas o en un tipo específico de contenido sin necesidad de reentrenar el modelo completo.

Los modelos MoE pueden ser entrenados de manera más efectiva en diferentes subconjuntos de datos, lo que facilita el ajuste del modelo para tareas específicas o para mejorar el rendimiento en ciertos contextos de traducción.

Dado que solo se activan algunos expertos durante el entrenamiento, el modelo puede ser entrenado de manera más eficiente. Esto reduce el tiempo y el costo asociados con el entrenamiento de modelos grandes y complejos.

2.9.1 ST-MoE

El trabajo de Zoph et al. (2022) presenta una arquitectura para modelos de expertos dispersos (*sparse expert models*) que se enfoca en la estabilidad y la capacidad de transferencia. La arquitectura ST-MoE se basa en un modelo de expertos dispersos que combina varios componentes clave:

1. **Modelo de Expertos Dispersos (MoE):** En esta arquitectura, el modelo se divide en varios "expertos" o submodelos especializados, cada uno entrenado para manejar diferentes aspectos del problema. Solo una fracción de estos expertos se activa en cada paso de inferencia, lo que hace que el modelo sea más eficiente.
2. **Combinador de Expertos:** Un componente importante de ST-MoE es el combinador, que decide qué expertos se activarán para una entrada dada. Este componente utiliza

mecanismos de atención o *gating* para seleccionar los expertos más relevantes, lo que ayuda a mantener la eficiencia y a reducir la complejidad computacional.

3. **Estabilidad y Transferibilidad:** El paper introduce técnicas para mejorar la estabilidad durante el entrenamiento y la capacidad de transferencia a nuevas tareas o dominios. Esto se logra mediante la regularización y técnicas específicas para ajustar los pesos de los expertos de manera que el modelo sea robusto a variaciones en los datos.
4. **Arquitectura Híbrida:** ST-MoE puede combinarse con diferentes tipos de redes neuronales, como redes neuronales profundas o transformadores, para aprovechar sus fortalezas mientras mantiene la eficiencia y la capacidad de adaptación.

La arquitectura ST-MoE busca abordar desafíos comunes en los modelos de expertos dispersos, como la inestabilidad durante el entrenamiento y la dificultad para transferir conocimientos a nuevas tareas, proporcionando una solución que mejora tanto la eficacia como la flexibilidad del modelo.

2.9.2 Switch Transformers

Un enfoque interesante surge del trabajo de Fedus et al. (2022)al introducir los modelos Switch Transformers y su capacidad para escalar eficientemente a modelos con billones de parámetros utilizando técnicas de dispersión (*sparsity*) simples y eficientes.

Uno de los aspectos destacados de este trabajo es el uso de MoE (*Mixture of Experts*), o Mezcla de Expertos, una técnica que permite mejorar la eficiencia y el rendimiento de los modelos de gran escala. Los modelos MoE son una forma de estructurar redes neuronales donde diferentes expertos (subredes especializadas) se activan de manera selectiva según el tipo de entrada o tarea específica, permitiendo una gestión más eficiente de los recursos computacionales y una mejor adaptabilidad a diferentes tipos de datos.

En el contexto de los *Switch Transformers*, los MoE juegan un papel crucial al permitir que el modelo active solo un subconjunto de expertos relevantes para una tarea particular, lo que reduce significativamente la carga computacional en comparación con enfoques donde todos los parámetros están activos simultáneamente. Esto no solo mejora la eficiencia computacional, sino que también puede llevar a mejores resultados de rendimiento al permitir una mejor especialización y adaptabilidad del modelo a diferentes partes del espacio de entrada.

Por lo tanto, el uso de MoE en *Switch Transformers* no solo es relevante para la escalabilidad a modelos con un gran número de parámetros, sino que también destaca cómo la estructuración inteligente y la gestión de recursos pueden ser críticas para avanzar en la capacidad de procesamiento y el rendimiento de los modelos de inteligencia artificial a gran escala.

Vale la pena retomar el concepto de MoE (Zhou, 2012) y detallar sobre las diferencias entre el enfoque clásico de ensemble. Los modelos MoE (*Mixture of Experts*) y los modelos Ensemble son enfoques distintos utilizados en *machine learning* para mejorar el rendimiento y la robustez de los modelos, pero difieren significativamente en su estructura y funcionamiento:

En un MoE, el modelo está compuesto por múltiples submodelos (llamados expertos) que son especializados en diferentes partes del espacio de entrada o en diferentes tareas específicas. Durante la inferencia, en lugar de activar todos los expertos simultáneamente, se utiliza una red de gate (o "interruptor") que decide qué expertos deben ser activados en función de la entrada actual. Esto permite que el modelo se adapte dinámicamente a diferentes características de los datos o a diferentes partes de una tarea compleja. Mejora la eficiencia computacional al no tener que procesar todos los expertos para cada entrada, y permite una mejor adaptabilidad y especialización del modelo.

Por otro lado, los *Ensemble Models* combinan múltiples modelos base (que pueden ser del mismo tipo o diferentes) para mejorar el rendimiento general del sistema. Cada modelo base en el Ensemble puede ser entrenado de manera independiente y luego sus predicciones se combinan mediante alguna estrategia de agregación (como votación, promedio, etc.).

Los MoE se diferencian de los enfoques clásicos de *Ensemble Models* pues se estructura un solo modelo con múltiples expertos activados selectivamente, mientras que Ensemble combina modelos independientes. Por otro lado, MoE utiliza recursos de manera más eficiente al activar selectivamente expertos, mientras que Ensemble requiere mantener y ejecutar múltiples modelos base por separado. MoE permite una adaptación dinámica a diferentes condiciones o partes del espacio de entrada, mientras que Ensemble combina estáticamente las predicciones de modelos preentrenados.

Si bien existen aplicaciones recientes muy interesantes tales como MoE-Mamba (Pióro et al., 2024) y MoEUT (Csordás et al., 2024), se excluyen del alcance del presente trabajo.

2.10 Space-State Models

Los modelos de espacio de estados son un marco matemático utilizado para describir el comportamiento de un sistema mediante un conjunto de ecuaciones diferenciales de primer orden. En este enfoque, el estado del sistema se representa mediante un vector de variables, y la dinámica del sistema se describe a través de cómo cambian estas variables de estado a lo largo del tiempo.

En términos más específicos:

1. **Estado del Sistema**: Se representa con un vector de variables (generalmente llamado x), que encapsula toda la información necesaria sobre el sistema en un momento dado.
2. **Ecuaciones Diferenciales**: Las dinámicas del sistema se describen con una ecuación diferencial de primer orden que muestra cómo cambia el vector de estado (x) a lo largo del tiempo. Esta ecuación suele tener la forma:

$$\frac{\mathrm{dx}}{\mathrm{dt}} = Ax + Bu$$

donde A es una matriz que describe las interacciones internas del sistema, B es una matriz que describe cómo las entradas u afectan el estado, y u es el vector de entrada del sistema.

3. **Salida del Sistema**: Además, el modelo también puede incluir una ecuación que describe cómo se obtiene la salida del sistema (y\mathbf{y}y) a partir del estado y las entradas:

$$y = Cx + Du$$

donde C y D son matrices que definen la relación entre el estado y la salida, y entre la entrada y la salida, respectivamente.

Los modelos estructurados de secuencia en el espacio de estados (S4) son una clase reciente de modelos de secuencia para el aprendizaje profundo que están ampliamente relacionados con las RNN, las CNN y los modelos clásicos de espacio de estados. Los autores Gu et al. (2021) abordan el desafío de modelar eficientemente secuencias largas al introducir espacios de estado estructurados.

Este enfoque innovador permite reducir la complejidad computacional al seleccionar de manera inteligente subconjuntos de estados relevantes para cada paso de tiempo en una secuencia. Al optimizar el procesamiento de información en secuencias extensas, el modelo no solo mejora la

eficiencia computacional, sino que también mantiene una capacidad robusta para capturar dependencias a largo plazo y estructuras complejas en los datos.

La capa S4 (Gu et al., 2021) realiza una transformación secuencia-a-secuencia no lineal, mapeando una secuencia de entrada a una secuencia de salida. Esta capa contiene HHH SSMs independientes de "entrada única y salida única" (SISO) con estados de dimensión NNN. Cada SSM de S4 se aplica a una dimensión de la secuencia de entrada, produciendo una transformación lineal independiente por canal de entrada. Luego se aplica una función de activación no lineal y, finalmente, una capa de mezcla lineal por posición para combinar las características independientes y generar la secuencia de salida.

La capa S4 utiliza el marco HiPPO (Gu et al., 2020) para la aproximación de funciones en línea, inicializando las matrices de estado con una matriz HiPPO, lo que ha demostrado mejorar el rendimiento. Aunque la matriz HiPPO-LegS no es diagonalizable de forma estable, se puede representar en una forma diagonal más baja (DPLR), lo que S4 utiliza para derivar una forma eficiente del núcleo de convolución.

La implementación eficiente de la capa S4 depende del contexto: en modo recurrente para generación en línea y en modo convolucional para secuencias completas. En el modo convolucional, la recurrencia lineal se representa como una convolución unidimensional, aprovechando transformadas rápidas de Fourier (FFTs) para paralelizar el proceso.

Cada capa S4 tiene parámetros entrenables, incluidos los parámetros de los SSMs y los de la capa de mezcla. Para cada SSM, se utilizan matrices de entrada, transición, salida y *feedthrough*, y se aplican parámetros de escala temporal para discretizar el tiempo continuo. La notación compacta para los estados y salidas de los SSMs se usa para simplificar la representación matemática.

La capa S5 (Smith et al., 2022) sustituye el banco de SSMs SISO (o el gran sistema bloque-diagonal) en S4 por un SSM MIMO con un tamaño de estado latente PPP y dimensiones de entrada y salida HHH. Esta versión discretizada del SSM MIMO se aplica a una secuencia de entrada vectorial u1:Lu_{1:L}u1:L, produciendo una secuencia de salidas del SSM (o preactivaciones) y1:Ly_{1:L}y1:L, utilizando estados latentes xkx_kxk. Luego, se aplica una función de activación no lineal para obtener la salida de la capa u1:L′u'_{1:L}u1:L′. A diferencia de S4, S5 no necesita una capa lineal adicional por posición, ya que las características están mezcladas de manera intrínseca. Además, el tamaño de

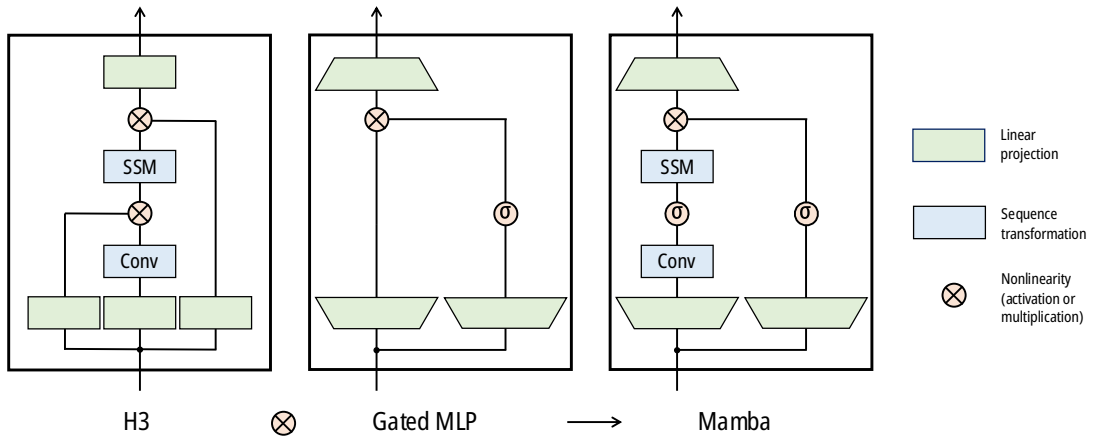
estado latente PPP de S5 puede ser significativamente menor que el tamaño HNHNHN del SSM bloque-diagonal en S4, permitiendo el uso de escaneos paralelos eficientes.

2.10.1 Mamba

Por su parte, Gu & Dao (2024) introducen innovaciones significativas en la modelización de secuencias al proponer un enfoque que mejora la eficiencia computacional y la capacidad de captura de dependencias a largo plazo en secuencias en una arquitectura denominada Mamba.

Una de las mejoras clave propuestas por Mamba es la introducción de espacios de estado selectivos (*Selective State Spaces*). Esto implica que en lugar de manejar todas las posibles combinaciones de estados en una secuencia de manera exhaustiva, el modelo Mamba selecciona dinámicamente un subconjunto relevante de estados para procesar en cada paso de tiempo. Esta selección se realiza de manera inteligente utilizando métodos de atención o *gating*, lo que permite al modelo concentrarse en las partes críticas de la secuencia y omitir el procesamiento de estados menos informativos o redundantes.

Al reducir el número de estados considerados en cada paso de tiempo, Mamba logra reducir drásticamente la complejidad computacional, permitiendo así el procesamiento en tiempo lineal respecto a la longitud de la secuencia. Esto es crucial para aplicaciones que manejan secuencias largas o requieren tiempos de respuesta rápidos.



Como lo describe Lieber et al. (2024) el diseño de bloque simplificado que se utilizando combina el bloque H3, que es la base de la mayoría de las arquitecturas de modelos de espacio de estados (SSM), con el bloque MLP ubicuo de las redes neuronales modernas. En lugar de entrelazar estos dos bloques, simplemente se repite el bloque Mamba de manera homogénea. En comparación

con el bloque H3, Mamba reemplaza la primera compuerta multiplicativa con una función de activación. En comparación con el bloque MLP, Mamba añade un SSM a la rama principal y para la activación, se utiliza la función SiLU (también conocida como Swish).

A pesar de la selección de estados, Mamba conserva la capacidad de modelar dependencias complejas y a largo plazo en las secuencias. Esto se logra mediante la adaptación dinámica de los estados seleccionados en función de la historia de la secuencia y las características del problema específico.

El enfoque de espacios de estado selectivos permite una mayor flexibilidad en la adaptación del modelo a diferentes tipos de secuencias y contextos. Puede ajustarse según las características específicas de la aplicación y las propiedades de las secuencias de datos.

Finalmente Vardasbi et al. (2023b) discute la aplicación del modelo S4 (*Structured State Spaces for Sequences*) en la traducción automática. Este modelo, que ha mostrado éxito en tareas de modelado de secuencias, como visión y audio, tiene la ventaja de comprimir la entrada en un único estado oculto, capturando dependencias a largo plazo sin necesitar mecanismos de atención.

Sin embargo, en el ámbito de la traducción automática, S4 no rinde tan bien como los modelos *Transformer*, mostrando una disminución de aproximadamente 4 puntos BLEU en su desempeño, especialmente con oraciones largas. La principal limitación identificada es la incapacidad del S4 para resumir la oración completa en un solo estado oculto. El estudio propone que la introducción de un mecanismo de atención en el modelo S4 podría cerrar esta brecha de rendimiento.

2.10.2 Simplified Structured State Space Sequence (S5)

Por su parte, Smith et al. (2022) presenta una arquitectura innovadora para el modelado de secuencias, centrada en la simplificación de las capas de espacio de estados.

La arquitectura se basa en la idea de simplificar las capas de espacio de estados (*state space layers*) para el modelado de secuencias. Los espacios de estados se utilizan para modelar dependencias a largo plazo en secuencias, y la simplificación busca hacer estas capas más eficientes y efectivas. Al simplificar las capas, se busca mejorar la capacidad del modelo para manejar secuencias largas y complejas de manera más eficiente.

La arquitectura simplifica las operaciones matemáticas complejas típicamente asociadas con los modelos de espacio de estados. Esto incluye la reducción de la complejidad computacional y la mejora de la estabilidad numérica. La simplificación permite que el modelo sea más fácil de implementar y más rápido de entrenar, sin sacrificar la capacidad de modelado.

El enfoque está diseñado para mejorar el rendimiento en tareas de modelado de secuencias, como el procesamiento de lenguaje natural (NLP) y la predicción de series temporales.

La arquitectura facilita el aprendizaje de dependencias a largo plazo en las secuencias, mejorando la capacidad del modelo para capturar patrones complejos.

La arquitectura propuesta por Smith et al. (2022) es compatible con redes neuronales profundas, lo que permite su integración con otras técnicas y modelos existentes en el campo del aprendizaje profundo.

Se puede utilizar en combinación con otros tipos de capas y arquitecturas, ofreciendo flexibilidad en el diseño de modelos.

2.10.3 Gated State Spaces

2.10.3.1 Modelos de Espacio de Estados y Dependencias a Largo Plazo:

Modelos de Espacio de Estados (SSM): Son modelos matemáticos que utilizan variables de estado para describir la dinámica de un sistema a lo largo del tiempo. Son eficaces para modelar secuencias en las que las dependencias abarcan largos períodos.

Dependencias a Largo Plazo: En los datos secuenciales (como textos o códigos), ciertos elementos pueden depender de elementos distantes en la secuencia. Por ejemplo, para entender una palabra en una oración, a veces es necesario conocer el contexto desde el inicio de la oración.

2.10.3.2 Modelado Autoregresivo de Secuencias:

Modelos Autoregresivos: Predicen el siguiente valor en una secuencia basado en los valores anteriores. En el contexto de datos secuenciales, generan o predicen secuencias paso a paso.

De acuerdo con Mehta et al. (2022) se explora el uso de modelos de espacio de estados para el modelado de dependencias a largo plazo en secuencias. Los modelos de espacio de estados (SSM) son herramientas matemáticas que emplean variables de estado para describir la dinámica temporal de un sistema, lo que permite el modelado de secuencias en las que las dependencias se extienden a

lo largo de períodos prolongados. Esta capacidad resulta particularmente útil en datos secuenciales, como textos o códigos, donde ciertos elementos pueden depender de otros situados a distancias considerables en la secuencia. Por ejemplo, comprender una palabra en una oración puede requerir considerar el contexto desde el inicio de la oración.

En el contexto del modelado autoregresivo de secuencias, los modelos autoregresivos se utilizan para predecir el siguiente valor en una secuencia en función de los valores anteriores, generando o prediciendo secuencias de manera secuencial. Esta técnica es crucial en aplicaciones de procesamiento del lenguaje y análisis de código.

El trabajo sobre *Gated State Spaces (GSS)* de Mehta et al. (2022) presenta varios avances significativos en el modelado de secuencias largas, particularmente en el contexto de tareas autoregresivas. Uno de los aportes principales es la mejora en la eficiencia del modelo. Tradicionalmente, los modelos basados en atención, como los Transformadores, tienen una complejidad cuadrática con respecto a la longitud de la secuencia, lo que implica un costo computacional prohibitivo para secuencias largas. En contraste, GSS reduce esta complejidad de $O(L2)$ a $O(L\log L)$ mediante el uso de la Transformada Rápida de Fourier (FFT) para realizar las convoluciones, lo que permite manejar secuencias de gran longitud de manera mucho más eficiente.

Además de la reducción en la complejidad, otro aspecto destacable de GSS es su capacidad de paralelización. Mientras que muchos modelos autoregresivos no pueden ser completamente paralelizados debido a la naturaleza secuencial del proceso de predicción, GSS es completamente paralelizable en la dimensión de longitud. Esta propiedad mejora significativamente la eficiencia en el entrenamiento, permitiendo un procesamiento más rápido en comparación con modelos como los basados en redes neuronales recurrentes (RNNs).

Una de las innovaciones clave que distingue a GSS es el uso de unidades de activación con puertas, o *gating units*, que ya habían demostrado ser útiles en otros contextos. Estas unidades permiten reducir la dimensionalidad de las operaciones más costosas, como las FFT, lo que acelera el entrenamiento sin comprometer la capacidad del modelo para capturar dependencias de largo alcance. Esta idea, que se basa en observaciones previas sobre la efectividad de las capas de alimentación con puertas, se extiende en GSS al contexto de los modelos de espacio de estado, lo que contribuye a una reducción significativa en el tiempo de entrenamiento sin afectar negativamente el rendimiento.

Un aspecto particularmente interesante de GSS es su simplicidad comparada con modelos anteriores como S4 y DSS. Mientras que S4, por ejemplo, dependía de una inicialización cuidadosa basada en matrices HiPPO para obtener buenos resultados, GSS elimina esta complejidad. Los autores del trabajo encontraron que el modelo no es tan sensible a la inicialización, lo que permite entrenarlo con inicializaciones aleatorias sin que se vea afectado su rendimiento. Esta simplificación contrasta con la dependencia de S4 y DSS en trucos de álgebra lineal numérica para garantizar su funcionamiento.

En cuanto a su relación con trabajos previos, GSS está profundamente vinculado con modelos como S4 y DSS. El modelo GSS se basa en la estructura de espacio de estado diagonal que S4 popularizó, y también aprovecha la eficiencia computacional que brinda la convolución mediante FFT. No obstante, a diferencia de S4, que presentaba cuellos de botella en el tiempo de entrenamiento, especialmente en hardware especializado como las TPUs, GSS logra superar estas limitaciones mediante un diseño más optimizado. Por otro lado, el modelo Mamba, que también aborda el uso de espacios de estado diagonalizados, introduce ciertas mejoras, pero GSS va un paso más allá al simplificar algunos de los aspectos más complejos del proceso, lo que resulta en un modelo más rápido en su entrenamiento. Un aspecto notable de la capa GSS es su capacidad para generalizar a entradas más largas o nuevos tipos de datos sin necesidad de entrenamiento adicional, lo que se conoce como generalización *zero-shot*. Esta característica resulta especialmente valiosa en aplicaciones prácticas donde los datos pueden variar con el tiempo.

Finalmente, se incorpora el mecanismo de autoatención para modelar las dependencias locales dentro de las secuencias. La combinación de autoatención con la capa GSS mejora la capacidad del modelo para capturar tanto las dependencias locales como las de largo plazo. La simplicidad en la implementación de la capa GSS la convierte en una opción práctica para investigadores y profesionales, facilitando su integración en sistemas de modelado de secuencias.

2.10.4 Jamba

Fuera del alcance de la presente comparativa, avances posteriores como Jamba (Lieber et al., 2024) representa un avance significativo al integrar dos enfoques innovadores en modelos de lenguaje: *Transformer* y *Mamba*. Este trabajo combina lo mejor de ambos mundos para mejorar tanto

la eficiencia computacional como la capacidad de modelado en el procesamiento de secuencias de texto largas.

En primer lugar, el modelo *Transformer* es conocido por su capacidad para capturar relaciones a largo plazo en secuencias mediante mecanismos de atención. Sin embargo, debido a su estructura densa y completa conectividad, los Transformers pueden ser computacionalmente costosos cuando se aplican a secuencias muy largas o a grandes volúmenes de datos.

Por otro lado, Mamba introduce los espacios de estado selectivos, lo que permite reducir la complejidad computacional al seleccionar dinámicamente un subconjunto de estados relevantes para cada paso de tiempo en una secuencia. Esto mejora la eficiencia del modelo al tiempo que mantiene su capacidad para modelar dependencias complejas a lo largo de la secuencia.

Jamba combina estas dos técnicas de manera híbrida. Utiliza la estructura del Transformer para capturar relaciones a largo plazo y la eficiencia de los espacios de estado selectivos de Mamba para manejar eficazmente secuencias largas. Esta combinación permite que el modelo Jamba mejore tanto la velocidad de procesamiento como la capacidad de modelado, haciendo posibles aplicaciones prácticas en el procesamiento de lenguaje natural donde tanto la eficiencia como la precisión son críticas.

2.10.5 MoE Mamba

Por otro lado, MoE-Mamba (Pióro et al., 2024) combina dos enfoques innovadores en modelado de secuencias: Mixture of Experts (MoE) y Mamba, que utiliza espacios de estado selectivos para mejorar la eficiencia y la capacidad de modelado. MoE permite gestionar de manera eficiente recursos computacionales al activar selectivamente expertos especializados según la tarea o contexto, mientras que Mamba optimiza el procesamiento al seleccionar dinámicamente estados relevantes en secuencias largas. Esta combinación no solo mejora la escalabilidad de los modelos al reducir la complejidad computacional, sino que también preserva la capacidad de capturar dependencias a largo plazo en secuencias complejas.

2.11 Moving Average Equipped Gated Attention

Por su parte, MEGA (*Moving Average Equipped Gated Attention*) es un mecanismo de atención de una sola cabeza que incluye un promedio móvil exponencial (EMA) para abordar dos limitaciones

del *Transformer* tradicional: la falta de sesgo inductivo y la complejidad cuadrática en la secuencia. Los aspectos principales de MEGA incluyen:

- **Inductivo posicional:** Al incorporar EMA, MEGA introduce un sesgo inductivo que captura dependencias locales, algo que la atención tradicional del Transformer no hace.
- **Complejidad:** MEGA es capaz de reducir la complejidad cuadrática del Transformer a complejidad lineal mediante la división de secuencias largas en "chunks" (bloques) con pérdida mínima de información contextual.
- **Rendimiento superior:** En experimentos con tareas de modelado de secuencias, como traducción automática, clasificación de imágenes y modelado de lenguaje, MEGA mostró mejoras significativas en comparación con Transformers y otros modelos recientes (Ma et al., 2022).
- **Diseño unificado:** MEGA está diseñado para ser eficiente tanto en tareas de dependencia de corto como largo alcance, abordando problemas de diversas modalidades de datos (lenguaje, audio, imagen y video) (Ma et al., 2022).

MEGA se diferencia de un Transformer "vainilla" en varios aspectos clave:

2.11.1.1 Atención con sesgo inductivo frente a atención sin sesgo inductivo:

MEGA incorpora un promedio móvil exponencial (EMA), lo que le permite modelar dependencias locales de manera más efectiva al introducir un sesgo inductivo posicional. Este sesgo ayuda a capturar dependencias entre tokens cercanos en una secuencia de forma más eficiente.

Transformers vainilla (el Transformer original de Vaswani et al.) no tienen un sesgo inductivo fuerte, lo que significa que la relación entre los tokens se aprende completamente de los datos, sin suposiciones previas sobre su proximidad o estructura posicional.

2.11.1.2 Complejidad computacional

MEGA mejora la eficiencia al reducir la complejidad cuadrática del cálculo de la atención en Transformers vainilla a una complejidad lineal. Esto se logra al dividir las secuencias largas en "chunks" o bloques fijos, lo que permite procesar secuencias más largas con menor costo computacional.

Los Transformers tienen una complejidad cuadrática en relación con la longitud de la secuencia debido al cálculo de las matrices de atención, lo que los hace ineficientes para secuencias largas (Vaswani et al., 2017).

2.11.1.3 Mecanismo de atención

MEGA utiliza un mecanismo de atención de una sola cabeza con compuertas (gated attention), lo que le permite ser tan expresivo como la atención de múltiples cabezas de un Transformer vainilla, pero con una arquitectura más simple.

Los Transformers (Vaswani et al., 2017) dependen de la atención multi-cabeza, lo que significa que múltiples atenciones se calculan en paralelo para capturar diferentes relaciones entre los tokens. Aunque es flexible, aumenta la complejidad computacional y la necesidad de recursos.

2.11.1.4 Manejo de dependencias locales y globales:

MEGA está diseñado específicamente para capturar tanto dependencias de corto como de largo alcance de manera más eficiente mediante el uso del EMA y las compuertas en su mecanismo de atención.

Los Transformers (Vaswani et al., 2017) tienen dificultades para capturar dependencias locales debido a su enfoque de atención global, lo que puede llevar a problemas cuando se trabaja con secuencias largas.

Ma et al. (2022) proponen un mecanismo de atención con medias móviles y puertas (Mega) para abordar simultáneamente dos debilidades. La idea principal es incorporar sesgos inductivos en el mecanismo de atención a lo largo de la dimensión temporal, aprovechando el enfoque clásico de la media móvil exponencial (EMA). La EMA captura dependencias locales que decrecen exponencialmente con el tiempo (ver Figura 1) y ha sido ampliamente utilizada en la modelación de datos de series temporales (§2). Se introduce una forma multidimensional atenuada de EMA con coeficientes ajustables (§3.1) y, a continuación, se desarrolla el mecanismo de atención con medias móviles y puertas integrando la EMA con una variante de la atención con una sola cabeza (Hua et al., 2022) (§3.2). Teóricamente, se demuestra que la atención con una sola cabeza es tan expresiva como la atención con múltiples cabezas, que es la más comúnmente utilizada (§3.3). Aprovechando el mecanismo de media móvil incorporado, se propone además una variante de Mega con complejidad

lineal, denominada Mega-chunk, que segmenta las secuencias de entrada en bloques fijos con una pérdida mínima de información contextual (§3.5).

Experimentalmente, a través de cinco tareas de modelado de secuencias en diversos tipos de datos, incluyendo modelado de secuencias con contexto largo, traducción automática neuronal, modelado de lenguaje autoregresivo, y clasificación de imágenes y voz, se demuestra que Mega supera significativamente a una variedad de modelos de referencia fuertes, tanto en términos de efectividad como de eficiencia (§4) (ver Tabla 1). Estas mejoras ilustran la importancia de modelar dependencias a largo y corto plazo mediante diferentes patrones de sesgos inductivos.

2.12 Extended Long Short-Term Memory

Los autores Beck et al. (2024) proponen xLSTM (*Extended Long Short-Term Memory*), es una innovación en redes neuronales recurrentes diseñada para mejorar el modelado de dependencias a largo plazo en secuencias temporales.

Este enfoque extiende la arquitectura estándar de LSTM mediante nuevas técnicas y capas adicionales que mejoran la captura y procesamiento de información a lo largo de secuencias largas. xLSTM supera las limitaciones de las LSTM convencionales al integrar avanzados mecanismos de atención y adaptabilidad, permitiendo una mejor representación de relaciones temporales complejas y contextos de larga duración en datos secuenciales. Este avance es crucial en aplicaciones donde la precisión temporal y la capacidad para modelar dependencias complejas son esenciales, como en predicción de series temporales o procesamiento de lenguaje natural.

Para abordar cómo mejorar el modelado de secuencias mediante LSTMs escalados a miles de millones de parámetros, se introducen técnicas como el *gating* exponencial con técnicas de normalización y estabilización adecuadas. Además, se modifica la estructura de memoria de LSTM, dando lugar a variantes como sLSTM con memoria y actualización escalares, y mLSTM que es completamente paralelizable con una estructura de memoria matricial y reglas de actualización de covarianza. Estas extensiones de LSTM se integran en bloques residuales que forman arquitecturas xLSTM, las cuales se apilan de forma residual. El uso de gating exponencial y estructuras de memoria modificadas potencia las capacidades de xLSTM, permitiéndole competir favorablemente con Transformers y Modelos de Espacio de Estados en términos de rendimiento y escalabilidad.

3 OBJETIVOS Y METODOLOGÍA DE TRABAJO

3.1 Objetivo general

Evaluar y comparar varias nuevas arquitecturas avanzadas posteriores a los Transformers para entender sus fortalezas y sus limitaciones en tareas de traducción neuronal automática.

3.2 Objetivos específicos

- Evaluar el rendimiento de diferentes arquitecturas de NMT posteriores a Transformers en términos de calidad de traducción y eficiencia computacional.
- Analizar las características clave de cada arquitectura en términos de capacidad de aprendizaje, escalabilidad y generalización.
- Identificar casos de uso específicos donde cada arquitectura pueda sobresalir, considerando diferentes idiomas y dominios de aplicación.

3.3 Metodología de trabajo

3.3.1 Tipo de investigación

Para llevar a cabo una investigación propedéutica que compare soluciones de Neural Machine Translation (NMT), es esencial establecer un enfoque metodológico riguroso que siga directrices claras, tal como señalan Leedy y Ormrod (2013). A menudo, se corre el riesgo de confundir la investigación documental con una mera recopilación de información o con la documentación superficial de hechos. Sin embargo, la investigación debe entenderse como un proceso lógico y consistente de recolección, análisis e interpretación de datos, cuyo objetivo es generar un conocimiento profundo sobre el fenómeno en estudio, en este caso, las soluciones de NMT.

Dado que se busca comparar diversas tecnologías de traducción automática basadas en redes neuronales, el enfoque metodológico debe seguir un proceso sistemático. Este proceso incluye la definición clara del objetivo de la investigación, la gestión adecuada de los datos, y la comunicación precisa de los hallazgos dentro de los marcos establecidos por la comunidad científica. Dichos marcos no solo guiarán sobre qué información incluir, sino también cómo realizar la comparación y qué tipos de conclusiones pueden extraerse de los datos obtenidos.

Para esta investigación, se adoptará un enfoque de investigación aplicada de tipo mixto (cualitativo y cuantitativo). Este enfoque es especialmente adecuado para el análisis de sistemas de NMT, ya que permite combinar el análisis estadístico de la precisión y el rendimiento de las distintas soluciones (enfoque cuantitativo), con una exploración más profunda de aspectos como la usabilidad, adaptabilidad y capacidad de aprendizaje de cada tecnología (enfoque cualitativo). De esta manera, se obtiene una visión más completa del impacto y las fortalezas de cada solución.

Aunque el enfoque mixto será el principal método de investigación, no se excluyen otros enfoques complementarios, como la investigación descriptiva. La investigación descriptiva permitirá observar y documentar el estado actual de cada solución de NMT, identificando características clave y posibles correlaciones entre el rendimiento de los sistemas y los contextos de uso. Este análisis descriptivo es crucial para evaluar de manera objetiva las capacidades de los modelos de traducción automática y su aplicabilidad en distintos escenarios.

En conclusión, siguiendo los lineamientos de una investigación estructurada, se llevará a cabo una comparativa exhaustiva de las soluciones de NMT, utilizando un enfoque mixto que permita una evaluación tanto cuantitativa como cualitativa de los sistemas. Esta metodología asegurará la validez y la utilidad de los hallazgos, proporcionando una base sólida para tomar decisiones informadas sobre qué solución de NMT es la más adecuada según los requerimientos específicos del entorno en estudio.

3.3.2 Fuentes

En el contexto de esta investigación comparativa sobre soluciones de Neural Machine Translation (NMT), las fuentes de información juegan un papel fundamental al proporcionar los datos y la evidencia necesarios para sustentar las evaluaciones y recomendaciones. Las fuentes de información incluyen todos aquellos recursos que contienen datos formales, estudios previos, material multimedia y otros formatos relevantes, que permiten construir una base sólida para el análisis y la argumentación científica. Estas fuentes no solo facilitan la recolección de información crítica, sino que también garantizan que las comparaciones entre las soluciones de NMT se realicen de manera rigurosa y fundamentada.

Las fuentes primarias utilizadas en esta investigación están compuestas por material bibliográfico y documental de alto prestigio y relevancia en el ámbito de la inteligencia artificial,

procesamiento de lenguaje natural y sistemas de traducción automática. Las principales fuentes incluyen:

- *IEEE Computer Society Digital Library:* Proporciona acceso a una vasta colección de artículos técnicos, investigaciones y conferencias especializadas en informática, inteligencia artificial y aprendizaje automático. Esta fuente es vital para obtener estudios recientes y análisis comparativos de algoritmos de NMT.
- *ACM Digital Library:* Una de las bibliotecas digitales más importantes en el campo de la informática y las ciencias computacionales. Aquí se encuentran artículos que tratan de avances recientes en NMT, evaluaciones de modelos y estudios experimentales que son cruciales para la comparativa de soluciones.
- *Cornell University arXiv:* Un repositorio de acceso abierto que ofrece preprints de investigaciones en diversas áreas de la ciencia y la tecnología. arXiv es particularmente útil para acceder a estudios de vanguardia y trabajos no publicados oficialmente, permitiendo a los investigadores mantenerse al tanto de los avances más recientes en traducción automática neuronal sin las barreras del acceso por suscripción.

El acceso a estas fuentes es clave para obtener la documentación necesaria. Las dos primeras fuentes mencionadas (IEEE y ACM Digital Library) requieren suscripción y membresía, lo que garantiza el acceso a contenido exclusivo y de alta calidad, lo que resulta crucial para asegurar la profundidad y precisión en el análisis. En cambio, arXiv, como una fuente de acceso libre, complementa este material al ofrecer investigaciones emergentes sin costo alguno. Esta combinación de recursos pagos y gratuitos permite abarcar un espectro amplio de conocimientos, desde los estudios más consolidados hasta las innovaciones más recientes en el campo de la NMT.

En todos los casos, se asegura el respeto a la autoría original de los trabajos consultados, reconociendo y citando adecuadamente a los autores. Esto no solo mantiene la integridad académica de la investigación, sino que también contribuye a construir un cuerpo de conocimiento basado en fuentes confiables y verificadas. Al utilizar estas fuentes primarias, la investigación se basa en datos sólidos y actualizados, lo que permite una comparación más precisa y fundamentada de las diversas soluciones de NMT evaluadas.

3.4 Estudio preliminar del estado del arte

Con el objetivo de proponer un tema de investigación en el área de Neural Machine Translation (NMT), se llevó a cabo una revisión preliminar de la literatura especializada, como parte de la propedéutica necesaria. Esta revisión permitió obtener una comprensión clara del estado actual del conocimiento, identificar las áreas previamente investigadas y detectar vacíos o cuestiones no resueltas que podrían ser abordadas en una nueva investigación. Siguiendo un proceso metódico, la revisión preliminar estableció las bases para el desarrollo de un tema sólido y relevante, facilitando la formulación de un problema de investigación pertinente y de los objetivos que guiarán el estudio.

3.4.1 Pasos realizados para la revisión preliminar de la literatura

3.4.1.1 Definición del tema de interés

En primer lugar, se delimitó claramente el tema de investigación, centrado en la comparación de soluciones de Neural Machine Translation. Se precisó el enfoque hacia la evaluación de aspectos como la precisión de la traducción, el rendimiento en diferentes idiomas y el impacto de las mejoras recientes en las arquitecturas de redes neuronales.

3.4.1.2 Identificación de fuentes de información relevantes

Posteriormente, se identificaron fuentes confiables, actualizadas y pertinentes. Para ello, se consultaron artículos académicos, libros, conferencias especializadas, tesis doctorales y estudios técnicos publicados en plataformas reconocidas como IEEE Computer Society Digital Library, ACM Digital Library y Cornell University arXiv.

3.4.1.3 Realización de la búsqueda bibliográfica

Se efectuó una búsqueda exhaustiva utilizando palabras clave como "Neural Machine Translation", "deep learning en NMT", "comparación de modelos NMT" y "traducción automática". La consulta en bases de datos especializadas como Scopus, Google Scholar y las bibliotecas digitales mencionadas permitió acceder a estudios relevantes y de calidad.

3.4.1.4 Selección de las fuentes más relevantes

Una vez obtenidas las referencias, se filtraron las fuentes mediante la lectura de resúmenes y secciones clave de los artículos, seleccionando aquellos estudios más pertinentes para los fines de la investigación y descartando información redundante o de escasa relevancia.

3.4.1.5 Análisis de la información recopilada

Se procedió al análisis detallado de las fuentes seleccionadas, identificando tendencias en la investigación, vacíos de conocimiento y problemas no resueltos. El análisis se enfocó especialmente en la comparación entre distintas arquitecturas neuronales (Transformers, LSTMs, modelos híbridos), en la efectividad de las mejoras recientes (como los modelos preentrenados) y en los retos asociados a la traducción de idiomas de baja difusión.

3.4.1.6 Formulación del problema de investigación

Con base en el análisis realizado, se formuló el problema de investigación, de manera específica, clara y relevante. En este caso, el problema se centró en la evaluación comparativa de varias soluciones de NMT, considerando su rendimiento en la traducción de textos complejos o multilingües, así como su eficiencia computacional y capacidad de adaptación a distintos contextos lingüísticos.

3.4.1.7 Establecimiento de los objetivos de investigación

Finalmente, se establecieron los objetivos de la investigación, definiéndolos como específicos, medibles y realistas.

La revisión preliminar de la literatura resultó ser un paso esencial para el planteamiento del tema de investigación en el campo de NMT. Gracias a este proceso sistemático, fue posible comprender el estado actual del área, identificar vacíos relevantes y formular preguntas de investigación pertinentes, sentando así una base sólida para el desarrollo de un estudio que aporte nuevos conocimientos a la comunidad académica y científica en el ámbito de la traducción automática neuronal.

3.5 Métodos de investigación

En el contexto de un trabajo universitario que busca comparar soluciones de Neural Machine Translation (NMT), es esencial emplear una variedad de técnicas investigativas formales para obtener resultados robustos y confiables. Estas técnicas permiten evaluar las diferentes bibliotecas y modelos de NMT de manera rigurosa. A continuación, se describen y explican las técnicas más relevantes para realizar una comparativa efectiva:

3.5.1 Experimentos Controlados

Los experimentos controlados son una técnica fundamental en la investigación que implica manipular y controlar variables específicas para evaluar el impacto de estas variables en los resultados obtenidos.

En el caso de la comparativa de soluciones de NMT, se pueden diseñar experimentos donde se varíen factores como la biblioteca de NMT utilizada (por ejemplo, OpenNMT, MarianNMT, o Fairseq), el hardware (diferentes configuraciones de GPU o TPU), y los conjuntos de datos (diversos corpora de traducción). La clave es planificar estos experimentos de manera que cualquier diferencia en el rendimiento se pueda atribuir a la variable manipulada, manteniendo constantes todas las demás. Esto asegura que los resultados sean válidos y fiables.

3.5.2 Estudios de Casos

Los estudios de casos implican la investigación detallada de escenarios específicos en los que se han utilizado diferentes soluciones de NMT en contextos reales. Esta técnica permite analizar cómo diferentes bibliotecas de NMT han sido implementadas en aplicaciones prácticas. Por ejemplo, se pueden examinar casos de uso en empresas de traducción automática, plataformas de servicios en línea, o aplicaciones en la industria de contenido multilingüe. Al recopilar y analizar datos de estos escenarios reales, se obtiene una visión sobre la eficacia y eficiencia de cada biblioteca en condiciones prácticas.

3.5.3 Análisis Comparativo

El análisis comparativo implica la evaluación cuantitativa y cualitativa de diferentes soluciones utilizando métricas específicas para identificar diferencias significativas. Para comparar bibliotecas de NMT, se deben recopilar datos sobre métricas clave como la precisión de la traducción, el tiempo de entrenamiento, y el uso de recursos (como la memoria y el procesamiento). Utilizando estos datos, se realiza una comparación entre las soluciones para determinar cuál ofrece el mejor rendimiento en términos de eficiencia y calidad de las traducciones. Este análisis puede incluir tanto comparaciones numéricas como evaluaciones cualitativas de la calidad de las traducciones generadas.

3.5.4 Revisiones Sistemáticas

Las revisiones sistemáticas son una técnica que implica la recopilación exhaustiva y el análisis crítico de la literatura existente sobre un tema específico. En una revisión sistemática, se busca y se analiza literatura relevante, como artículos científicos, documentos técnicos, y estudios de caso sobre bibliotecas de NMT. Este proceso incluye la búsqueda exhaustiva en bases de datos académicas y técnicas, la aplicación de criterios de inclusión y exclusión para seleccionar estudios relevantes, y el análisis crítico para identificar tendencias actuales, avances tecnológicos y bibliotecas destacadas en el campo.

3.5.5 Validación Cruzada

La validación cruzada es una técnica que evalúa la capacidad de generalización de un modelo al probarlo en diferentes subconjuntos de datos. Para evaluar el rendimiento de las bibliotecas de NMT, se utilizan técnicas de validación cruzada. Esto implica dividir un conjunto de datos en varias particiones, entrenar y validar los modelos en estas particiones, y luego evaluar su desempeño en términos de precisión de traducción y eficiencia. La validación cruzada ayuda a asegurar que los modelos no estén sobreajustados a un único conjunto de datos y puedan generalizar bien a diferentes contextos y tipos de datos.

4 PLANTEAMIENTO DE LA COMPARATIVA

4.1 Criterios de selección de los conjuntos de datos

La elección adecuada de los datos para realizar una comparativa de soluciones de traducción automática neuronal (NMT) es crucial porque los datos tienen un impacto directo en la calidad de la evaluación y en la fiabilidad de los resultados obtenidos. A continuación, se detallan los aspectos clave que destacan la importancia de seleccionar correctamente los conjuntos de datos:

4.1.1 Representatividad de los Datos

Es esencial que los datos utilizados en la comparativa sean representativos del entorno real donde se espera utilizar el modelo NMT. Si los datos no cubren adecuadamente el rango de lenguajes, dialectos, temas y estilos lingüísticos esperados, los resultados de la comparativa no reflejarán el

desempeño real del sistema en producción. Por ejemplo, un modelo entrenado con textos académicos puede no rendir bien en un contexto informal o conversacional.

4.1.2 Diversidad Lingüística

Los datos deben cubrir un amplio espectro de pares de idiomas, especialmente si se evalúan soluciones multilingües. Los diferentes idiomas presentan desafíos únicos, como diferencias en la estructura gramatical, las expresiones idiomáticas, y la complejidad morfológica. Incluir un conjunto diverso de lenguas asegura que el modelo pueda manejar adecuadamente las particularidades de cada una y permite evaluar su versatilidad y robustez.

4.1.3 Variedad en Estilos y Dominios

Es importante que los datos cubran varios estilos de texto (formal, informal, técnico, literario) y dominios (finanzas, medicina, entretenimiento, etc.). Los modelos NMT pueden tener un rendimiento diferente según el tipo de contenido que traduce. Evaluar los modelos solo con datos de un dominio específico puede sesgar los resultados y dar una idea equivocada de su capacidad para generalizar a otros contextos.

4.1.4 Calidad de los Datos de Entrenamiento y Pruebas

La calidad de los datos es clave. Un corpus con traducciones de baja calidad o mal alineadas entre los idiomas afectará tanto el entrenamiento como la evaluación. Para una evaluación precisa de los modelos, es importante contar con conjuntos de prueba que contengan traducciones de alta calidad realizadas por expertos humanos, que sirvan como referencia confiable (gold standard) para medir el rendimiento de las soluciones NMT.

4.1.5 Tamaño del Conjunto de Datos

El tamaño del conjunto de datos de entrenamiento también influye en los resultados. Modelos entrenados con grandes cantidades de datos tienden a tener un mejor rendimiento, ya que tienen acceso a más ejemplos para aprender patrones y estructuras lingüísticas. Sin embargo, para garantizar una comparativa justa, todos los modelos deben entrenarse con conjuntos de datos de tamaños similares o ajustarse mediante técnicas de regularización para evitar ventajas desiguales.

4.1.6 Datos Equilibrados y Balanceados

Es fundamental que los datos estén equilibrados para evitar que los modelos se sesguen hacia ciertos pares de lenguajes, dominios o estilos. Si el conjunto de datos contiene una desproporción en el número de ejemplos entre diferentes idiomas o contextos, los modelos tenderán a rendir mejor en los casos con más ejemplos, lo que falsearía los resultados de la comparativa.

4.1.7 Datos Actualizados

El lenguaje es dinámico y cambia con el tiempo. Es importante usar datos que estén actualizados para reflejar el estado actual de los lenguajes y no quedarse obsoletos. El uso de datos antiguos puede llevar a resultados que no representen correctamente la capacidad del modelo para manejar nuevas terminologías, modismos o cambios en la estructura lingüística.

4.1.8 Conjunto de Pruebas Adecuado

Un conjunto de pruebas adecuado debe ser lo suficientemente diverso y extenso como para evaluar con precisión las capacidades de los modelos en diferentes escenarios. Es crucial evitar la contaminación del conjunto de pruebas (es decir, que los datos de entrenamiento incluyan partes del conjunto de pruebas), ya que esto podría sesgar los resultados y dar una impresión falsa del rendimiento del modelo.

4.1.9 Problemas de sesgo y justicia

Es importante seleccionar los datos de manera que no se introduzcan sesgos que afecten a ciertos grupos o lenguajes. Un conjunto de datos con un sesgo cultural o lingüístico podría llevar a modelos NMT que reflejen esas mismas deficiencias, lo que puede tener implicaciones negativas cuando se implementen en aplicaciones del mundo real.

4.1.10 Tamaño y Complejidad del Texto

El tamaño y la complejidad del conjunto de datos son factores clave. Un conjunto de datos muy sencillo puede no ser un desafío suficiente para evaluar con precisión la capacidad de un modelo. Es esencial que los datos incluyan frases complejas, variaciones en la longitud de las oraciones y estructuras gramaticales desafiantes para que la comparativa refleje el rendimiento del modelo en un espectro amplio de dificultades lingüísticas.

4.2 Corpus paralelo

Un corpus paralelo se diferencia de otros tipos de corpus en que contiene textos que están alineados y son traducciones directas entre dos o más idiomas. Esto significa que cada documento en un corpus paralelo tiene una contraparte directa en otro idioma, lo cual es fundamental para entrenar y evaluar sistemas de traducción automática y para realizar investigaciones comparativas entre idiomas.

Los corpus paralelos son relativamente escasos por varias razones:

- **Costo y tiempo de creación:** La creación de un corpus paralelo requiere recursos significativos en términos de tiempo, dinero y esfuerzo humano. La recopilación y alineación manual de textos paralelos pueden ser laboriosas y costosas.
- **Necesidad de derechos de autor:** Obtener los derechos para traducir y distribuir textos en varios idiomas puede ser complicado y costoso. Esto puede limitar la disponibilidad de textos paralelos.
- **Calidad y consistencia:** Es crucial que las traducciones en un corpus paralelo sean precisas y de alta calidad para que los datos sean útiles en aplicaciones de procesamiento de lenguaje natural. Esto implica una cuidadosa revisión y validación de los textos paralelos.
- **Diversidad de dominios y géneros:** Es deseable que los corpus paralelos cubran una amplia gama de dominios (por ejemplo, noticias, literatura, textos técnicos) y géneros (como diálogos, narrativas, instrucciones), lo cual puede ser difícil de conseguir de manera exhaustiva.

Debido a estas razones, la construcción y disponibilidad de corpus paralelos de alta calidad y tamaño suficiente para diversos propósitos de investigación y desarrollo tecnológico puede ser limitada, lo que contribuye a que sean considerados recursos valiosos pero escasos en el ámbito del procesamiento de lenguaje natural.

4.2.1 UN Parallel Corpus V1

El Corpus Paralelo de las Naciones Unidas (Ziemski et al., 2016) es un recurso lingüístico invaluable compilado por las Naciones Unidas. Consiste en textos paralelos en múltiples idiomas, es decir, textos que son versiones traducidas del mismo contenido.

Este corpus es particularmente útil para tareas como la traducción automática, modelado del lenguaje e investigación cruzada lingüística. Investigadores y desarrolladores lo utilizan

frecuentemente para entrenar y evaluar sistemas de tecnología del lenguaje. Su disponibilidad contribuye significativamente al avance de las capacidades de procesamiento de lenguaje natural multilingüe.

Cuadro No. 4.1. Características del conjunto de datos UN Parallel Corpus V1 (UNPC)

Espacio requerido:	Número total de líneas:	Vocabulario EN (tokens):	Vocabulario ES (tokens):	Número de líneas utilizadas:
~3,81 GB	25 227 004	577 671 824	669 899 929	250 000

Fuente: https://opus.nlpl.eu/UNPC/en&es/v1.0/UNPC

4.2.2 OpenSubtitles v2018

El dataset OpenSubtitles v2018 (Lison & Tiedemann, 2016) es una colección de subtítulos de películas que ha sido recopilada y distribuida como un recurso lingüístico ampliamente utilizado en la investigación de procesamiento de lenguaje natural y aprendizaje automático.

OpenSubtitles v2018 contiene subtítulos de películas en múltiples idiomas. Los subtítulos están alineados, lo que significa que hay correspondencias directas entre las líneas de diálogo en diferentes idiomas cuando las películas se han traducido.

Cuadro No. 4.2. Características del conjunto de datos OpenSubtitles (OSPC)

Espacio requerido:	Número total de líneas:	Vocabulario EN (tokens):	Vocabulario ES (tokens):	Número de líneas utilizadas:
~3.80 GB	61 434 251	391 976 667	364 279 696	250 000

Fuente: https://opus.nlpl.eu/OpenSubtitles/en&es/v2018/OpenSubtitles

Es uno de los datasets más grandes disponibles públicamente para el entrenamiento y evaluación de modelos de traducción automática y otros sistemas de procesamiento de lenguaje natural. Contiene millones de líneas de diálogo en diversos idiomas.

Debido a su tamaño y variedad lingüística, OpenSubtitles v2018 es utilizado para una variedad de aplicaciones de investigación, incluyendo la traducción automática, la generación de subtítulos automáticos, la investigación en lingüística computacional y más.

El dataset está disponible gratuitamente para la comunidad de investigación, lo cual lo hace accesible para investigadores, desarrolladores y entusiastas que trabajan en el campo del procesamiento de lenguaje natural.

4.2.3 Ventajas y desventajas de los conjuntos de datos seleccionados

En una comparativa de soluciones de traducción automática neuronal (NMT) para el par de lenguas inglés-español (EN-ES), elegir el corpus adecuado es fundamental para obtener resultados precisos y realistas. A continuación, se comparan las ventajas de usar UN Parallel Corpus V1 y OpenSubtitles v2018 en este contexto, destacando cómo cada uno puede beneficiar o limitar el rendimiento del modelo en diferentes aspectos (P. Lin et al., 2024).

4.2.3.1 Contenido y estilo lingüístico

UN Parallel Corpus V1: Este corpus está compuesto principalmente discursos y documentos formales traducidos por expertos humanos. Su contenido se centra en temas de política, derecho internacional, desarrollo y relaciones diplomáticas. Por lo tanto, el estilo es formal, técnico y altamente especializado, lo que lo hace adecuado para entrenar modelos NMT que se utilizarán en entornos formales o técnicos. Entre las ventajas del uso de este conjunto de datos están:

- Ideal para sistemas NMT que se empleen en traducción técnica o formal, como documentos legales, gubernamentales o de negocios.
- Alta calidad de las traducciones, ya que los documentos oficiales son revisados por traductores profesionales.
- El lenguaje es consistente y estandarizado, facilitando la generalización en estos dominios.

Sin embargo, UN Parallel Corpus presente algunas desventajas como las siguientes:

- No es adecuado para tareas de traducción cotidiana o informal debido al estilo altamente especializado y formal.
- Puede no reflejar la variabilidad en el uso del lenguaje cotidiano, lo que afecta su capacidad para manejar textos coloquiales o informales.

Por su parte OpenSubtitles v2018 está compuesto por subtítulos de películas y programas de televisión. Los subtítulos abarcan una gran variedad de géneros, como comedia, drama, acción y

documentales. El estilo de lenguaje es informal y conversacional, con una mezcla de registros coloquiales y expresiones idiomáticas. Entre las ventajas del uso de este conjunto de datos están:

- Excelente para entrenar modelos NMT en contextos informales o conversacionales, como aplicaciones de chat, redes sociales o traducción de contenido audiovisual.
- Mayor diversidad en el vocabulario y estilo, incluyendo coloquialismos, expresiones idiomáticas, e incluso lenguaje técnico en algunos géneros de cine.
- Refleja un uso más variado y actual del idioma, lo que lo hace ideal para aplicaciones de uso general.

Sin embargo, OpenSubtitles v2018 presenta algunas desventajas como las siguientes:

- La calidad de las traducciones puede ser inconsistente, ya que los subtítulos son muchas veces realizados por aficionados o no se ajustan a los estándares de traducción profesional.
- Puede contener errores o simplificaciones, lo que podría afectar la precisión del modelo si se utiliza para tareas formales.

4.2.3.2 Dominio de Aplicación

El conjunto de datos UN Parallel Corpus V1 es adecuado para entrenar modelos orientados a la traducción en entornos especializados como agencias gubernamentales, instituciones internacionales, o empresas que necesitan traducciones precisas y formales. Sin embargo, su uso limita la capacidad del modelo para traducir textos de otros dominios (por ejemplo, literatura, conversación diaria), ya que el vocabulario es más técnico y menos adaptable a contextos coloquiales.

Por su parte OpenSubtitles v2018 permite entrenar modelos que son más versátiles y adaptables a diferentes géneros y registros lingüísticos, como aplicaciones de traducción para consumidores, subtitulado de videos y servicios de traducción en línea. Sin embargo, no es adecuado para aplicaciones que requieren traducción formal o especializada, ya que el estilo del lenguaje es mucho más coloquial y podría no captar matices técnicos o formales con la precisión necesaria.

4.2.3.3 Tamaño del Corpus y Cobertura de Idiomas

En el caso del dataset UN Parallel Corpus V1 aunque su tamaño es más limitado en comparación con OpenSubtitles, ofrece un contenido altamente enfocado en temas oficiales, con traducciones cuidadosas y uniformes en cuanto a calidad. Pero debido a su enfoque especializado, la

cantidad de datos puede ser insuficiente para entrenar modelos grandes que necesiten generalizar en dominios más amplios.

Por otro lado, OpenSubtitles v2018 es significativamente más grande que el UN Parallel Corpus, lo que lo convierte en un recurso ideal para modelos que requieren un gran volumen de datos para aprender patrones diversos. Al abarcar millones de líneas de subtítulos, proporciona una gran variedad de ejemplos y contextos. El volumen de datos no siempre está alineado con la calidad, ya que puede haber ejemplos mal traducidos o inconsistentes.

4.2.3.4 Diversidad Cultural y Contextualización

El dataset UN Parallel Corpus V1 tiene un enfoque cultural y lingüístico más neutral, siendo apropiado para traducciones donde se requiere mantener un tono imparcial y estándar en diferentes contextos internacionales. Puede carecer de la flexibilidad contextual para manejar diferencias culturales en el uso cotidiano del lenguaje.

Una de las mayores ventajas de OpenSubtitles v2018 es la amplia gama de temas, personajes y situaciones en los subtítulos permite que el modelo capture una mayor diversidad cultural y de uso contextual del lenguaje. Sin embargo, esta diversidad también introduce más ruido en los datos, lo que podría desorientar al modelo en tareas que requieran consistencia estilística o formal.

4.2.3.5 Calidad de Traducción

UN Parallel Corpus V1 ofrece traducciones de alta calidad, revisadas por profesionales, lo que garantiza que el modelo NMT se entrene con pares de frases precisas y consistentes. La falta de variabilidad en la calidad de las traducciones puede limitar la exposición del modelo a diferentes estilos o formas de traducción.

OpenSubtitles v2018 contiene una mezcla de traducciones profesionales y no profesionales, lo que ofrece un entrenamiento más variado en términos de calidad. Esto puede hacer que el modelo sea más robusto en el manejo de diferentes niveles de traducción. La inconsistencia en la calidad puede introducir errores, especialmente si los datos no son filtrados adecuadamente.

UN Parallel Corpus V1 es ideal para entrenar y evaluar modelos NMT en contextos formales, técnicos o especializados, donde la precisión y consistencia son cruciales. Sin embargo, su aplicación es limitada en dominios más informales o generales. OpenSubtitles v2018 es más adecuado para

tareas de traducción cotidiana e informal, con un enfoque en el uso coloquial del idioma. Su gran volumen y diversidad ofrecen más flexibilidad, aunque a costa de una menor consistencia en la calidad de las traducciones.

En una comparativa de soluciones NMT enfocada en inglés-español, la elección del corpus dependerá del dominio y el propósito final del sistema de traducción. Idealmente, una combinación de ambos corpora podría ofrecer un balance entre precisión formal y versatilidad informal (P. Lin et al., 2024).

5 DESARROLLO DE LA COMPARATIVA

5.1 Arquitecturas seleccionadas

Como se revisió en la primera parte de la memoria, en el ámbito de Neural Machine Translation (NMT), varias arquitecturas avanzadas han surgido para mejorar el rendimiento y la eficiencia de los modelos de traducción. A continuación, se presenta una selección detallada de estas arquitecturas, destacando sus innovaciones y características clave:

5.1.1 Transformer Vainilla

Los transformers (Vaswani et al., 2017) introdujeron una arquitectura basada en mecanismos de atención, eliminando la necesidad de redes recurrentes (RNN) o convolucionales (CNN) en el procesamiento de secuencias. La atención auto-regresiva permite al modelo considerar todas las palabras en una secuencia de entrada simultáneamente, lo que mejora la capacidad de capturar dependencias a largo plazo. La atención escalada por producto punto, que permite el procesamiento paralelo y mejora la eficiencia en el manejo de secuencias largas.

5.1.2 Universal Transformer

El Universal Transformer (Dehghani et al., 2019) es una extensión del Transformer tradicional que introduce una recurrencia en el mecanismo de atención. Esta arquitectura permite que cada capa de la red realice múltiples pasadas sobre la entrada, ajustando su comportamiento a diferentes niveles de la secuencia.

La incorporación de la recurrencia en el mecanismo de atención, que permite un mejor manejo de secuencias de longitud variable y ajusta el nivel de atención a lo largo de la red.

5.1.3 ST-MoE

El trabajo de Zoph et al., 2022 sobre ST-MoE (Sparse Transformer Mixture of Experts) introduce una arquitectura que utiliza un conjunto de expertos para mejorar la eficiencia computacional. Solo una fracción de los expertos se activa para cada ejemplo, lo que reduce el costo computacional sin sacrificar la calidad de la traducción. La utilización de una mezcla de expertos dispersos que permite un entrenamiento más eficiente y reduce el tiempo de inferencia.

5.1.4 Mamba

Mamba (Gu & Dao, 2024) introduce una arquitectura innovadora de modelado secuencial que supera varias limitaciones fundamentales del Transformer. A diferencia de este, que escala cuadráticamente con la longitud de la secuencia, Mamba logra un rendimiento lineal gracias al uso de modelos de espacios de estado estructurados (SSMs) con mecanismos de selección dependientes del contenido. Esto le permite filtrar dinámicamente información irrelevante y preservar lo esencial según el contexto, sin recurrir a mecanismos de atención ni bloques MLP tradicionales. Además, está optimizado para hardware moderno, lo que se traduce en una inferencia hasta cinco veces más rápida que la de los Transformers. Su diseño simple y homogéneo no solo facilita la implementación, sino que también ofrece un rendimiento competitivo o superior en tareas de lenguaje, audio y genómica, igualando a Transformers del doble de tamaño en calidad de resultados y mejorando la generalización en secuencias extremadamente largas.

5.1.5 Gated State Spaces

Los Gated State Spaces (Mehta et al., 2022) utilizan puertas para controlar el flujo de información en el modelo, combinando conceptos de redes neuronales recurrentes con Transformers. Esta arquitectura busca mejorar el manejo de dependencias temporales y espaciales. La integración de puertas en el espacio de estado, que permite un manejo más flexible y efectivo de las dependencias a largo plazo.

5.1.6 Moving Average Equipped Gated Attention

Ma et al., 2022 introduce una arquitectura con un mecanismo de atención equipado con promedios móviles para mejorar la captura de patrones a largo plazo en las secuencias de entrada. La atención con promedios móviles ayuda a estabilizar el entrenamiento y a mejorar la precisión de la

traducción. El uso de promedios móviles en el mecanismo de atención, que mejora la capacidad del modelo para capturar dependencias a largo plazo y reduce la varianza en el entrenamiento.

Cuadro No. 5.1. Principales innovaciones de los modelos seleccionados para la comparativa.

Modelo	Innovaciones
Universal Transformer (UT)	Añade recurrencia en profundidad (repetidas refinaciones por posición), combinando paralelismo con inductivo recurrente; puede detener dinámicamente (*dynamic halting*) el refinamiento por token, mientras que Vainilla aplica una pila fija de capas. (Dehghani et al., 2019)
Gated State Space (GSS)	Sustituye atención global por modelos de espacio de estado más eficientes (O(L logL) en lugar de O(L\2)), con *gating* para reducir la dimensionalidad y acelerar FFTs; más eficiente en secuencias largas (Mehta et al., 2022).
Mamba	Elimina la atención y MLPs, usando modelos de espacio de estado selectivos basados en el input; logra razonamiento dependiente del contenido y es lineal en longitud; *Transformer* depende de self-attention fijo sin selección dinámica (Gu & Dao, 2024).
Mega	Reemplaza *multi-head attention* por una atención de una sola cabeza equipada con promedio móvil exponencial (EMA), introduciendo *bias* posicional local; ofrece variante de complejidad lineal mediante *chunking*, mientras *Transformer* usa *self-attention* con costos cuadráticos (Ma et al., 2022).
ST-MoE	Introduce *Mixture-of-Experts (MoE)* para activar dinámicamente solo partes del modelo, reduciendo cómputo efectivo mientras escala en tamaño total de parámetros; *Transformer* usa todos sus parámetros en cada paso. Además, aplica técnicas de estabilización y *fine-tuning* especializadas (Zoph et al., 2022).

5.2 Implementación y Experimentación

Para comparar estas arquitecturas, se debe implementar y entrenar cada una utilizando un conjunto de datos estándar para NMT. Los conjuntos de datos recomendados incluyen:

- UN Parallel Corpus (Ziemski et al., 2016): Un corpus multilingüe que proporciona datos paralelos en varios idiomas, adecuado para la evaluación de modelos de traducción automática.

- OpenSubtitles V2018 (Lison & Tiedemann, 2016): Un corpus que contiene subtítulos de películas y series en múltiples idiomas, ideal para evaluar la calidad de la traducción en contextos coloquiales y conversacionales.

5.3 Métricas de Evaluación

Las métricas estándar para evaluar la calidad de la traducción automática BLEU (Bilingual Evaluation Understudy) es una métrica que mide la precisión n-grama y compara la salida del modelo con traducciones de referencia.

5.4 Análisis Comparativo

El análisis comparativo debe abarcar:

- Calidad de Traducción: Evaluar la precisión y fluidez de las traducciones generadas por cada arquitectura utilizando métricas como BLEU.
- Tiempo de Entrenamiento: Medir el tiempo necesario para entrenar cada modelo, incluyendo el tiempo de configuración, entrenamiento y ajuste.
- Consumo de Recursos: Evaluar el uso de recursos computacionales, como la memoria y el procesamiento, para cada arquitectura.
- Capacidad de Generalización: Analizar cómo cada arquitectura maneja datos de prueba no vistos durante el entrenamiento y su rendimiento en diferentes conjuntos de datos.

La implementación y evaluación de estas arquitecturas avanzadas de NMT proporcionará una visión completa de sus capacidades y limitaciones. La comparación basada en la calidad de traducción, eficiencia, y recursos permitirá identificar las mejores prácticas y soluciones para tareas específicas de traducción automática. Este análisis detallado contribuirá a la comprensión de cómo las innovaciones recientes en NMT pueden mejorar el rendimiento y la eficiencia en aplicaciones prácticas.

5.5 Configuración del entorno

En una comparativa de soluciones para algoritmos de entrenamiento de redes neuronales, el *hardware* utilizado desempeña un papel fundamental en el rendimiento y la eficiencia del proceso de entrenamiento. El *hardware* adecuado puede marcar una gran diferencia en términos de velocidad de cálculo, capacidad de procesamiento paralelo y manejo eficiente de grandes volúmenes de datos. En

esta sección, describiremos detalladamente el *hardware* utilizado en el entorno de prueba, incluyendo las especificaciones de la CPU, GPU y la cantidad de memoria RAM disponible.

El tipo y modelo de la CPU juegan un papel crucial en el rendimiento general del entrenamiento de redes neuronales. Algunos modelos de CPU ofrecen instrucciones específicas para acelerar las operaciones matemáticas requeridas durante el entrenamiento de redes neuronales, lo que puede resultar en una mejora significativa en el tiempo de entrenamiento. Además, la capacidad de la CPU para manejar múltiples hilos de ejecución puede aprovecharse para realizar cálculos paralelos y agilizar el proceso.

La GPU, por otro lado, es un componente esencial cuando se trata de acelerar el entrenamiento de redes neuronales. Las GPU están diseñadas para realizar operaciones matemáticas de manera altamente paralela, lo que las convierte en una opción ideal para tareas intensivas en cómputo, como el entrenamiento de redes neuronales. La elección del modelo de GPU y su memoria de video disponible pueden influir directamente en la velocidad y capacidad de entrenamiento de la red neuronal.

Además, la cantidad de memoria RAM disponible es un factor crítico por considerar pues el tamaño del conjunto de datos y la arquitectura de la red neuronal determinarán la cantidad de memoria necesaria para almacenar los datos y los parámetros del modelo durante el entrenamiento. Una cantidad insuficiente de memoria RAM puede llevar a problemas de rendimiento, como la ralentización del entrenamiento o incluso la incapacidad de cargar y procesar conjuntos de datos grandes.

En el caso de esta comparativa se utilizó el siguiente *hardware* y *software*:

Cuadro No. 5.2. Configuración del entorno de prueba.

Elemento	Descripción
CPU	AMD Ryzen 9 5950X 4.9 Ghz 16 Cores / 32 Hilos Caché L3: 64 MB Enfriamiento líquido.
GPU	NVIDIA GeForce RTX 3060 12 GB de memoria GDDR6 y 3584 CUDA Cores.

Elemento	Descripción
Memoria	64 GB de Memoria DDR4 (4 módulos)
Almacenamiento	Kingston NV1 NVMe M.2 PCIe Gen3, 1TB.
Sistema operativo	WSL2 sobre Windows 11 Professional (Ubuntu 22.04.2 LST)

Nota: Se utilizaron dos computadoras configuradas idénticamente con el fin de realizar el análisis en el tiempo definido para esta comparativa de soluciones.

5.5.1 Habilitar compatibilidad con GPU

Se consideraron tres formas distintas de configurar *Pytorch* con soporte a tecnologías GPU:

- Sobre una distribución de Linux no virtualizada[3].
- Sobre un contenedor de Linux o máquina virtual.
- Sobre el subsistema de Linux en Windows (WSL2)[4].

Para esta investigación se toma la decisión de utilizar la opción de WSL2. Particularmente, WSL2 es una característica de Windows que permite a los usuarios ejecutar aplicaciones, contenedores y herramientas de línea de comandos nativas de Linux en Windows 11 y versiones posteriores del sistema operativo. El WSL 2 ofrece beneficios como compatibilidad total con aplicaciones de Linux, un sistema de archivos más rápido y una integración estrecha con el sistema operativo Windows.

El procedimiento descrito en la guía "*CUDA on WSL User Guide*" explica cómo utilizar la tecnología de NVIDIA CUDA en el Subsistema de Windows para Linux (WSL). La guía se centra en el uso de CUDA en el WSL 2, lo que permite aprovechar la aceleración por GPU de NVIDIA para aplicaciones de ciencia de datos, aprendizaje automático e inferencia en Windows a través del WSL.

A continuación, se resume el procedimiento:

- Instalar el controlador de NVIDIA necesario para habilitar el soporte de GPU,

[3] https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#package-manager-metas
[4] Véase https://docs.nvidia.com/cuda/wsl-user-guide/index.html

- Instalar y configurar el entorno de desarrollo de Linux en WSL. Es necesario crear un archivo de configuración "*.wslconfig"* para ampliar la cantidad de memoria que utiliza el subsistema Linux:

```
[wsl2]
memory=56GB # Limits VM memory in WSL 2 to 56 GB
swap=8GB
```

- Se deben instalar los componentes de software de NVIDIA, como los controladores de Windows, el kit de herramientas de CUDA y las bibliotecas adicionales.
- Existen algunas limitaciones y características no admitidas en el entorno de WSL 2, como el soporte de GPU limitado en ciertos modelos de GPU y la falta de compatibilidad con algunas características de CUDA.
- Instala WSL y elige una distribución basada en glibc, como Ubuntu o Debian.
- Configurar un entorno de Python dentro de WSL. Se recomienda utilizar *Miniconda* de Anaconda para configurar un entorno virtual de Python.
- Una vez que *Miniconda* esté instalado en WSL, se crea un entorno con el nombre de cada uno de los algoritmos que utilizan Python y deben instalarse de forma consistente todos los requisitos necesarios a manera de una línea base de configuración.
- Es importante descargar y "actualizar las últimas versiones de los componentes y bibliotecas de tiempo de ejecución con el soporte de *hardware* de la comparativa.

5.5.2 Línea base

La línea base de configuración de un entorno Python se refiere a la configuración inicial necesaria para establecer un entorno de desarrollo y ejecución adecuado para trabajar con Python y sus bibliotecas. A continuación, se enumeran algunos aspectos básicos de la configuración de un entorno Python:

- *Instalación de Python:* Lo primero que se debe hacer es instalar Python en el sistema, para esta comparativa se utiliza la versión estable de Python 3.8.
- *Gestor de paquetes:* Python cuenta con gestores de paquetes como *pip* o *conda*, que facilitan la instalación y gestión de bibliotecas y dependencias. Es importante tener un gestor de

paquetes actualizado y configurado correctamente para instalar y mantener las bibliotecas requeridas por el proyecto.

- *Entorno virtual:* Se recomienda utilizar entornos virtuales para aislar los proyectos y evitar conflictos entre las dependencias. Los entornos virtuales permiten tener instalaciones separadas de Python y sus bibliotecas para cada proyecto.
- *Editor de código:* Elegir un editor de código o un entorno de desarrollo integrado (IDE) adecuado es importante para desarrollar en Python en este caso se eligió *Visual Studio Code* por su fácil integración con el entorno WSL2.
- *Configuración del entorno de trabajo:* Dependiendo de las necesidades específicas del proyecto, se pueden configurar variables de entorno, como PATH, que permiten acceder a las herramientas y bibliotecas de Python desde cualquier ubicación del sistema.
- *Bibliotecas y dependencias:* Para trabajar con diferentes funcionalidades en Python, se necesitarán bibliotecas y dependencias específicas. Estas se pueden instalar utilizando el gestor de paquetes correspondiente (*pip* o *conda*) y se pueden especificar en un archivo de requisitos para facilitar la replicación del entorno.

Finalmente, dependiendo de los requisitos de cada algoritmo, pueden existir otras configuraciones adicionales, como configuraciones de parámetros globales del sistema operativo e instalar bibliotecas y componentes de tiempo de ejecución adicionales.

5.6 Aspectos específicos del dominio

En una comparativa de modelos de traducción automática neuronal (NMT, por sus siglas en inglés), es esencial considerar varios aspectos técnicos y arquitectónicos que influyen directamente en la calidad de las traducciones y la eficiencia del entrenamiento de los modelos. A continuación, exploraremos los factores clave que deben analizarse en una comparativa exhaustiva de soluciones NMT, más allá del hardware involucrado.

Las arquitecturas de los modelos NMT han evolucionado desde los enfoques basados en secuencias recurrentes (RNN) hacia redes más avanzadas como los Transformadores. Es importante destacar que los modelos basados en Transformadores, como BERT o GPT, han demostrado un rendimiento superior en la calidad de traducción y en la capacidad de manejar dependencias a largo

plazo. Comparar las diferentes arquitecturas ayuda a evaluar su eficiencia, precisión y capacidad de generalización.

Por su parte, el tamaño del modelo es otro aspecto crítico en la comparativa. Modelos más grandes con mayor número de parámetros tienden a captar mejor las relaciones complejas en los datos, pero requieren más recursos computacionales y memoria. En este sentido, es importante evaluar el trade-off entre el rendimiento de traducción y la cantidad de recursos necesarios para entrenar modelos más grandes.

Es esencial incluir métricas estandarizadas como BLEU (Bilingual Evaluation Understudy), METEOR o TER (Translation Edit Rate) para evaluar la calidad de las traducciones generadas por los diferentes modelos. Estas métricas permiten medir la cercanía entre la traducción automática y una traducción de referencia, proporcionando un indicador objetivo del rendimiento de cada solución NMT.

Además de la precisión, la velocidad con la que un modelo puede ser entrenado y la rapidez con la que produce traducciones son factores fundamentales, especialmente en aplicaciones de traducción en tiempo real. Aquí entra en juego no solo el tipo de hardware utilizado (como se mencionó anteriormente), sino también la optimización del software, los algoritmos de búsqueda utilizados durante la inferencia (por ejemplo, beam search o greedy search) y el uso de técnicas como el paralelismo de datos o de modelo.

En un entorno de comparativa de modelos NMT, es relevante considerar si el sistema puede gestionar eficientemente múltiples pares de lenguajes. Los modelos multilingües, como los de arquitectura basada en Transformadores, permiten entrenar una única red neuronal para varios idiomas, lo cual puede mejorar la eficiencia del sistema y reducir la complejidad, aunque a veces a costa de la precisión en algunos lenguajes específicos.

La capacidad de un modelo para generalizar a ejemplos no vistos durante el entrenamiento es crucial para determinar su viabilidad en entornos del mundo real. Evaluar la robustez de los modelos bajo diferentes condiciones (por ejemplo, presencia de ruido, frases ambiguas o variantes dialectales) es un aspecto clave en una comparativa de soluciones NMT.

Aunque el rendimiento es fundamental, la eficiencia en el uso de los recursos computacionales no debe pasarse por alto. Aquí, la elección del framework (por ejemplo, PyTorch) y

las técnicas de optimización (como el uso de cuantización o pruning) influyen en el rendimiento y en la capacidad de aprovechar mejor el hardware disponible.

Finalmente, la escalabilidad de una solución NMT es crucial en escenarios donde es necesario procesar grandes volúmenes de datos o donde se requiere una actualización frecuente de los modelos. Un buen sistema de NMT debe poder adaptarse a diferentes necesidades de escalabilidad sin sacrificar la calidad o la eficiencia. Incluir las características arquitectónicas, métricas de calidad, velocidad de entrenamiento, eficiencia y escalabilidad permite una evaluación más completa y ajustada a las necesidades del entorno en el que se va a desplegar el sistema.

5.7 Casos de prueba y experimentos

Aunque los conjuntos de datos UN Parallel Corpus V1 y OpenSubtitles V2018 son una referencia pública limitada y es patente que los conjuntos de datos a escala industrial son mucho más grandes en términos de muestras, el rendimiento de los métodos y algoritmos considerados en la comparativa en este conjunto de datos es representativo para aproximar la idoneidad de estos aplicados a problemas del mundo real.

Cuadro No. 5.3. Casos de prueba

Conjunto de datos	Número de ejemplos	Ejemplos para entrenamiento	Ejemplos para validación	Ejemplos para prueba
UN-Parallel Corpus V1[1]	250000 (2,19%)	175000 (~1.54%)	50250 (~0.44%)	24750 (~0,22%)
OpenSubtitles V2018	1000000 (1,62%)	700000 (~1.14%)	201000 (~0.33%)	99000 (~0.16%)

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

5.8 Hiperparámetros dependientes de los datos

En la evaluación y comparativa de soluciones de Neural Machine Translation (NMT), los hiperparámetros dependientes de los datos juegan un papel crucial en la configuración y rendimiento del modelo. Estos hiperparámetros afectan directamente cómo el modelo procesa y aprende de los datos, lo que, a su vez, impacta la calidad de la traducción, la eficiencia del entrenamiento y la capacidad de generalización. A continuación, se explica el efecto de cada uno de estos hiperparámetros en una comparativa de NMT:

5.8.1 Longitud Mínima de Secuencia

La longitud mínima de secuencia se refiere al tamaño más corto de las secuencias de entrada y salida que el modelo considera durante el entrenamiento.

- Calidad de Traducción: Si la longitud mínima de secuencia es demasiado corta, el modelo podría ignorar información contextual importante, lo que podría degradar la calidad de la traducción para textos más largos. Por otro lado, si es demasiado larga, puede incluir secuencias irrelevantes que dificulten el entrenamiento.
- Eficiencia del Entrenamiento: Configurar una longitud mínima adecuada ayuda a reducir el ruido y las secuencias demasiado cortas que pueden no aportar suficiente información para el aprendizaje.
- Capacidad de Generalización: Asegurarse de que la longitud mínima se ajuste a la naturaleza del conjunto de datos garantiza que el modelo pueda manejar secuencias de longitud variable y generalizar mejor en diferentes contextos.

5.8.2 Longitud Máxima de Secuencia

La longitud máxima de secuencia define el tamaño máximo de las secuencias que el modelo procesa durante el entrenamiento.

- Calidad de Traducción: Establecer una longitud máxima adecuada asegura que el modelo pueda capturar la información completa de las secuencias largas sin truncar partes importantes del texto. Una longitud máxima demasiado corta puede llevar a la pérdida de contexto y reducir la calidad de la traducción.
- Eficiencia del Entrenamiento: Secuencias más largas requieren más recursos computacionales, lo que puede aumentar el tiempo de entrenamiento. Establecer una longitud máxima adecuada ayuda a equilibrar la precisión del modelo y la eficiencia del entrenamiento.
- Capacidad de Generalización: Permitir secuencias de longitud variable asegura que el modelo pueda manejar diferentes tipos de datos y contextos, mejorando su capacidad para generalizar a textos de longitud diversa.

5.8.3 Tamaño del Vocabulario

El tamaño del vocabulario en un idioma define el número de palabras o tokens únicos que el modelo utiliza para representar el texto en un idioma.

- Calidad de Traducción: Un vocabulario demasiado pequeño puede causar problemas de cobertura, ya que palabras raras o especializadas podrían ser reemplazadas por un token desconocido (e.g., <UNK>). Un vocabulario más grande puede mejorar la calidad de la traducción al capturar una gama más amplia de términos y matices.
- Eficiencia del Entrenamiento: Vocabularios más grandes aumentan el tamaño del modelo y requieren más memoria y poder de cómputo para el entrenamiento y la inferencia. Un vocabulario muy grande también puede hacer que el modelo sea más lento y menos eficiente.
- Capacidad de Generalización: Un tamaño de vocabulario bien ajustado ayuda al modelo a generalizar mejor al manejar términos y expresiones variados, lo que puede ser especialmente útil en dominios especializados o lenguajes con riqueza léxica.

5.8.4 Batch Size (Tamaño del Lote)

El tamaño del lote se refiere al número de ejemplos de entrenamiento procesados simultáneamente en una sola iteración de entrenamiento.

- Calidad de Traducción: Un tamaño de lote más grande puede llevar a una convergencia más estable y rápida durante el entrenamiento, pero también puede resultar en una menor capacidad de generalización si el modelo no se ajusta adecuadamente a los datos.
- Eficiencia del Entrenamiento: Los tamaños de lote más grandes suelen ser más eficientes desde el punto de vista computacional, ya que permiten un procesamiento paralelo más eficaz en hardware moderno. Sin embargo, también requieren más memoria.
- Capacidad de Generalización: Tamaños de lote más pequeños a menudo proporcionan actualizaciones más ruidosas pero más frecuentes, lo que puede ayudar a evitar el sobreajuste y mejorar la capacidad de generalización del modelo. Ajustar el tamaño del lote es crucial para encontrar un equilibrio entre eficiencia y capacidad de generalización.

Cuadro No. 5.4. Hiperparámetros dependientes de los conjuntos de datos.

Conjunto de datos	Minimum Sequence Length	Maximum Sequence Length	EN Vocabulary Size	ES Vocabulary Size	Batch Size
UN-Parallel Corpus V1[1]	3	941	50288	68931	8
OpenSubtitles V2018	3	80	70729	94982	16

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

5.9 Convenciones

Cuando se comparan diferentes arquitecturas Encoder-Decoder para Neural Machine Translation (NMT), es fundamental estandarizar ciertos aspectos del entrenamiento y evaluación para garantizar que los resultados sean comparables y válidos. Las convenciones utilizadas en esta comparativa incluyen optimizadores, schedulers, early stopping, clipping, y medidas de pérdida. A continuación se explica cada una de estas convenciones en detalle:

5.9.1 Optimizador

Se utilizará ADAM con Tasa de Aprendizaje de 0.0001. El optimizador ADAM (*Adaptive Moment Estimation*) es ampliamente utilizado en el entrenamiento de redes neuronales debido a su capacidad para adaptar las tasas de aprendizaje para diferentes parámetros y manejar grandes volúmenes de datos de manera eficiente.

Efecto en la Comparación:

- Consistencia en el Entrenamiento: Utilizar ADAM con una tasa de aprendizaje fija de 0.0001 asegura que las comparaciones entre arquitecturas sean justas, ya que todas las arquitecturas se entrenan bajo las mismas condiciones de optimización.
- Rendimiento del Modelo: La tasa de aprendizaje de 0.0001 es una elección común que permite un entrenamiento estable y controlado, evitando problemas de convergencia demasiado rápida o lenta.

5.9.2 Scheduler

Se configura LROnPlateau que ajusta la tasa de aprendizaje cuando el rendimiento del modelo (por ejemplo, la medida de pérdida en el conjunto de validación) se estabiliza o empeora.

Efecto en la Comparación:

- Adaptación Dinámica: Permite que la tasa de aprendizaje se ajuste en respuesta a la falta de mejora en la pérdida del modelo, lo que puede ayudar a obtener una convergencia más eficiente y efectiva.
- Comparabilidad: Usar el mismo scheduler en todas las arquitecturas asegura que las decisiones de ajuste de tasa de aprendizaje no influyan en las diferencias observadas en el rendimiento entre modelos.

5.9.3 Early Stopping

El *Early stopping* detiene el entrenamiento si la medida de pérdida en el conjunto de validación no mejora durante un número específico de épocas. En este caso, el entrenamiento se detendrá si no hay mejora durante 5 épocas consecutivas.

Efecto en la Comparación:

- Prevención del Sobreentrenamiento: Reduce el riesgo de sobreajuste al detener el entrenamiento cuando no hay mejoras significativas, lo que asegura que los modelos no se entrenen en exceso.
- Equidad en la Comparación: Garantiza que todas las arquitecturas tengan el mismo límite para detener el entrenamiento, evitando que algunas arquitecturas tengan más oportunidades de ajustarse a los datos que otras.

5.9.4 Clipping

El *clipping* de gradientes limita el valor de los gradientes durante el entrenamiento a un máximo especificado para prevenir el problema del desbordamiento de gradientes, que puede ocurrir en redes profundas.

Efecto en la Comparación:

- Estabilidad del Entrenamiento: Evita que los gradientes excesivamente grandes causen problemas de inestabilidad durante el entrenamiento, lo que es crucial para arquitecturas complejas.

- Consistencia en el Rendimiento: Aplicar el mismo *clipping* a todas las arquitecturas asegura que los problemas de inestabilidad relacionados con los gradientes no influyan en las diferencias observadas en el rendimiento de las arquitecturas.

5.9.5 Medida de Pérdida

La medida de pérdida de entropía cruzada (*Cross-Entropy*) es una función de pérdida comúnmente utilizada para tareas de clasificación, incluyendo la traducción automática, que mide la diferencia entre las distribuciones de probabilidad predicha y real.

Efecto en la Comparación:

- Relevancia de la Medida: Cross-Entropy es una medida estándar para problemas de clasificación y es adecuada para evaluar la precisión de las predicciones en el contexto de NMT.
- Comparabilidad Directa: Utilizar Cross-Entropy como medida de pérdida permite comparar directamente el rendimiento de diferentes arquitecturas en términos de su capacidad para minimizar la discrepancia entre las predicciones y las traducciones reales.

Estas convenciones establecen una base uniforme para comparar diferentes arquitecturas Encoder-Decoder en NMT, asegurando que las diferencias en el rendimiento sean atribuibles a las características de las arquitecturas mismas y no a variaciones en el proceso de entrenamiento.

6 DISCUSIÓN Y ANÁLISIS DE RESULTADOS

En esta sección se discuten los resultados de los cuatro algoritmos de entrenamiento de redes neuronales considerados. Es importante destacar que se establece una comparativa entre el tiempo total del entrenamiento, el tamaño del modelo en memoria, la exactitud de los modelos (entrenamiento y validación) y la cantidad de parámetros entrenables de los modelos.

6.1 Cantidad de Parámetros

El tamaño del modelo es un factor crítico para considerar en escenarios donde el almacenamiento de recursos es limitado, como en dispositivos con capacidades de almacenamiento limitadas o entornos con restricciones de memoria. Un modelo con un tamaño más grande requerirá más recursos para su almacenamiento y carga en memoria durante el despliegue y la ejecución.

El análisis es relevante porque proporciona información sobre el tamaño promedio del modelo para cada algoritmo evaluado. El tamaño del modelo puede tener un impacto significativo en la practicidad y eficiencia de su implementación en diferentes entornos.

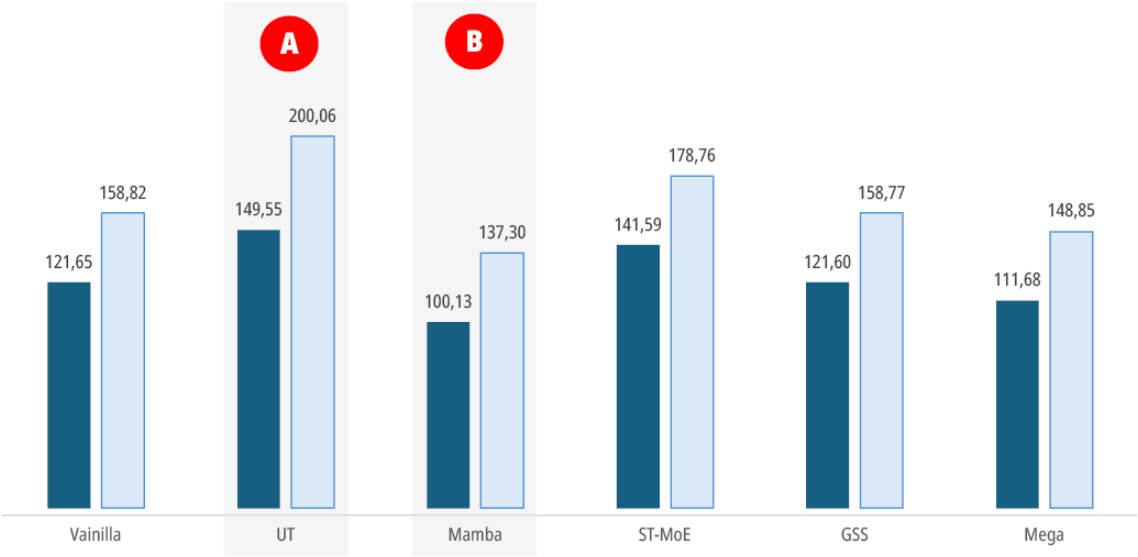
Cuadro No. 6.1. Cantidad de parámetros por modelo y conjunto de datos.

Modelo	UNPC	OSPC
Vainilla	121,65	158,82
UT	149,55	200,06
Mamba	100,13	137,30
ST-MoE	141,59	178,76
GSS	121,60	158,77
Mega	111,68	148,85

Nota: Cifras expresadas en millones de parámetros.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.1. Número de parámetros entrenables de los modelos de la comparativa.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La figura expone el número de parámetros entrenables de cada una de las arquitecturas evaluadas, expresado en millones (MM), y muestra cómo distintos enfoques afectan la escala del modelo, tanto en el corpus UNPC como en OSPC. Esta variable es fundamental en el análisis de modelos, ya que está directamente relacionada con la capacidad de generalización, el consumo de memoria y el coste computacional durante el entrenamiento y la inferencia.

El modelo Vainilla, con 121,65 millones de parámetros entrenables en el conjunto UNPC y 158,82 millones en OSPC, sirve como línea base para comparar la eficiencia y escalabilidad de las arquitecturas más recientes. La arquitectura Universal Transformer (UT) presenta un incremento sustancial en el número de parámetros, alcanzando 149,55 y 200,06 millones, respectivamente. Este crecimiento responde a la incorporación de mecanismos de recurrencia en las capas de atención, lo cual incrementa la capacidad del modelo a costa de una mayor complejidad computacional.

Por su parte, el modelo Mamba muestra una propuesta de eficiencia estructural: aunque reduce significativamente la cantidad de parámetros en UNPC (100,13 millones), mantiene una capacidad expresiva competitiva, como lo evidencia su valor en OSPC (137,30 millones). Esta característica sugiere una optimización del uso de parámetros sin comprometer el desempeño, tal como se destaca en el punto B de la gráfica.

La arquitectura ST-MoE (Sparse Transformer con Mixture-of-Experts) se distingue por una notable expansión paramétrica, especialmente en el corpus OSPC, donde alcanza los 178,76 millones de parámetros. Esta cifra es coherente con el enfoque MoE, que habilita rutas especializadas dentro del modelo mediante expertos activados selectivamente, lo cual incrementa la capacidad sin incrementar proporcionalmente el costo de inferencia por muestra.

En cuanto al modelo GSS, se observa una configuración muy cercana al Transformer original en términos de cantidad de parámetros (121,60 en UNPC y 158,77 en OSPC), lo que sugiere que las mejoras introducidas no incrementan sustancialmente la complejidad del modelo, pero podrían ofrecer beneficios cualitativos en términos de generalización o estabilidad del entrenamiento.

Finalmente, Mega aparece como una solución intermedia, con una arquitectura que reduce levemente la cantidad de parámetros en comparación con otras innovaciones, manteniendo 111,68 millones en UNPC y 148,85 millones en OSPC. Su desempeño paramétrico sugiere un compromiso entre eficiencia y expresividad, ubicándose como una alternativa balanceada frente al resto de arquitecturas evaluadas.

En síntesis, el análisis revela dos patrones relevantes: (A) algunas arquitecturas tienden a incrementar significativamente el número de parámetros para mejorar el desempeño, y (B) modelos como Mamba o Mega muestran que es posible mantener o incluso mejorar la calidad de traducción

con una reducción o uso más eficiente de los parámetros, lo que resulta especialmente relevante en entornos con recursos computacionales limitados.

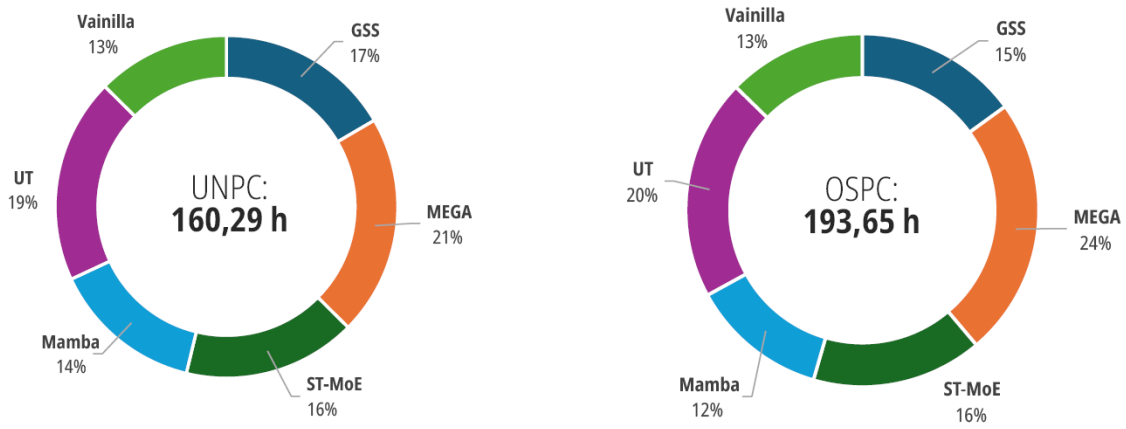
6.2 Tiempo de entrenamiento y validación

Además de la cantidad de parámetros entrenables, otro factor determinante en la evaluación de arquitecturas de traducción automática neuronal es el tiempo total requerido para completar el entrenamiento y la validación del modelo. Tener un tiempo de ejecución más bajo puede ser beneficioso en situaciones en las que se requiere una respuesta rápida o cuando hay restricciones de tiempo. Esto puede ser especialmente relevante en aplicaciones en tiempo real o en entornos donde se necesitan respuestas rápidas a consultas o predicciones.

Este análisis es importante porque proporciona información sobre el tiempo promedio de ejecución de cada algoritmo evaluado. Esta información es relevante ya que el tiempo de ejecución puede tener un impacto significativo en la practicidad y eficiencia de un algoritmo en un entorno de producción.

Las gráficas presentadas a continuación muestran el tiempo total de entrenamiento y validación requerido por distintas arquitecturas de modelos de traducción automática en dos conjuntos de datos: **UNPC** (United Nations Parallel Corpus) y **OSPC** (OpenSubtitles Parallel Corpus). El análisis revela diferencias significativas en la eficiencia computacional de las arquitecturas evaluadas, tanto en entornos con lenguaje formal (UNPC) como en lenguaje más informal y fragmentado (OSPC).

Gráfico No. 6.2. Contribución relativa al tiempo total de ejecución de la comparativa



Nota: Tiempo promedio dado en horas tras efectuar 20 epochs. Se utiliza UNPC para designar al conjunto de datos Corpus Paralelo de las Naciones Unidas (Ziemski et al., 2016) y OSPC para designar a OpenSubtitles v2018 (Lison & Tiedemann, 2016). Fuente: Elaboración propia del estudiante.

En el caso del corpus UNPC, con un tiempo total de entrenamiento de 160,29 horas, destaca el modelo MEGA como el más demandante, representando un 21 % del total. Le siguen UT (19 %) y GSS (17 %), mientras que Mamba logra una participación relativamente baja del 14 %, sugiriendo una mayor eficiencia temporal frente a otras arquitecturas más complejas. El modelo Vainilla, por su parte, se posiciona como uno de los más eficientes con solo un 13 % del tiempo total, lo que refuerza su carácter ligero y fácil de entrenar, aunque posiblemente con limitaciones en términos de rendimiento final.

El panorama es similar al utilizar el corpus OSPC, donde el tiempo total asciende a 193,65 horas, debido en parte a la mayor cantidad de registros, a pesar de su menor longitud promedio por secuencia. Nuevamente, MEGA es el modelo que mayor tiempo de cómputo requiere, con un 24 %, seguido por UT con un 20 %. En contraste, Mamba se consolida como la opción más rápida (12 % del tiempo), reafirmando lo observado anteriormente: es una arquitectura optimizada para tiempos de entrenamiento cortos sin sacrificar su competitividad en precisión.

Es importante destacar que tanto ST-MoE como GSS presentan tiempos intermedios (16 % y 15 %, respectivamente), lo cual es coherente con su diseño estructural: ST-MoE introduce especialización sin implicar entrenamiento completo de todos los expertos simultáneamente, y GSS, pese a sus ajustes sobre el modelo base, mantiene una complejidad similar al Transformer original.

En términos globales, el análisis de esta métrica permite establecer que modelos como Mamba ofrecen ventajas significativas en eficiencia temporal, haciéndolos especialmente adecuados

para aplicaciones donde los ciclos de entrenamiento deben ser breves o frecuentes. Por otro lado, modelos como UT o MEGA, aunque más pesados en términos de tiempo, pueden justificar su uso si demuestran mejoras notables en desempeño durante las etapas de inferencia o generalización.

Estos resultados subrayan la importancia de considerar el tiempo total de entrenamiento y validación como una métrica clave al comparar arquitecturas de modelos. Más allá de la calidad de traducción obtenida, la eficiencia en el uso de recursos computacionales puede ser decisiva en contextos reales de despliegue, especialmente cuando se trata de escalabilidad, costos de infraestructura y tiempos de iteración en el desarrollo de modelos.

6.2.1 Tiempo de entrenamiento

Al comparar arquitecturas como Mamba, Transformer, Universal Transformer (UT), MEGA y otras, es fundamental considerar no solo su precisión o capacidad de generalización, sino también el tiempo total que requieren para ser entrenadas. Este factor es clave porque refleja la eficiencia computacional del modelo en condiciones reales de uso, donde los recursos (tiempo, energía y hardware) son limitados. Dos modelos con rendimientos similares pueden diferir significativamente en los costos asociados a su entrenamiento si uno requiere muchas más horas de cómputo o mayor consumo de memoria.

Además, el tiempo de entrenamiento incide directamente en la viabilidad de escalar el modelo a conjuntos de datos más grandes o realizar iteraciones frecuentes durante el desarrollo. Por ello, el tiempo total de entrenamiento es una métrica crítica para evaluar el balance entre rendimiento y eficiencia de cada arquitectura.

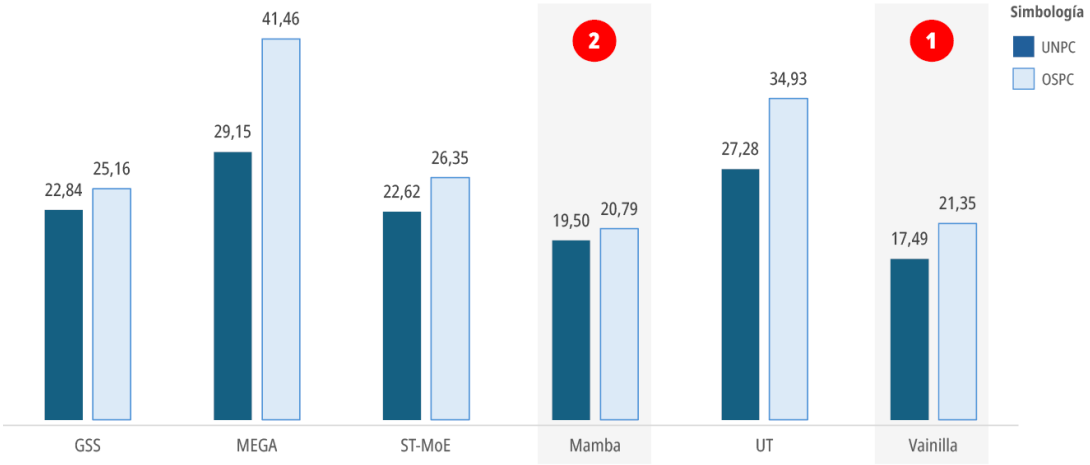
Cuadro No. 6.2. Tiempo total de la fase de entrenamiento por modelo

Modelo	UNPC	OSPC
GSS	22,84	25,16
MEGA	29,15	41,46
ST-MoE	22,62	26,35
Mamba	19,50	20,79
UT	27,28	34,93
Vainilla	17,49	21,35

Nota: Tiempo de entrenamiento en horas.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.3. Tiempo total de la fase de entrenamiento por modelo



Nota: Tiempo promedio dado en horas tras efectuar 20 epochs. Se utiliza UNPC para designar al conjunto de datos Corpus Paralelo de las Naciones Unidas (Ziemski et al., 2016) y OSPC para designar a OpenSubtitles v2018 (Lison & Tiedemann, 2016). Fuente: Elaboración propia del estudiante.

El cuadro y el gráfico presentados muestran el tiempo de entrenamiento total requerido por cada una de las arquitecturas evaluadas, en los conjuntos de datos **UNPC** (Corpus Paralelo de Naciones Unidas) y **OSPC** (OpenSubtitles), tras 20 épocas. Esta métrica permite observar con mayor precisión la eficiencia computacional específica de cada modelo, independientemente del tiempo destinado a la validación.

La eficiencia de un modelo de traducción automática no se limita únicamente al número total de parámetros o al desempeño final obtenido, sino que también debe evaluarse desde la perspectiva del tiempo de entrenamiento requerido por época, el cual tiene un impacto directo sobre los ciclos de iteración en investigación aplicada, los costos de entrenamiento en producción y la escalabilidad de las soluciones en entornos reales.

Al analizar los datos, se identifican diferencias sustantivas entre las arquitecturas evaluadas. El modelo Vainilla, correspondiente a la arquitectura original de Vaswani et al. (2017), es consistentemente el más rápido en ambas condiciones, con valores de 17,49 h (UNPC) y 21,35 h (OSPC), lo que refuerza su utilidad como baseline eficiente, especialmente en etapas tempranas de experimentación.

Por otro lado, el modelo Mamba destaca por ser la arquitectura optimizada más eficiente en esta métrica, con tiempos de entrenamiento cercanos al Vainilla (19,50 h en UNPC y 20,79 h en OSPC), incluso superándolo ligeramente en eficiencia en el corpus OSPC. Esto valida la hipótesis de que

Mamba puede ofrecer una excelente relación costo-beneficio en términos de tiempo y desempeño, como se ha evidenciado también en la sección de parámetros.

En contraste, las arquitecturas UT y MEGA son las que presentan los mayores tiempos de entrenamiento, alcanzando 27,28 h y 29,15 h respectivamente en UNPC, y hasta 34,93 h y 41,46 h en OSPC, lo cual refleja su complejidad estructural y la carga computacional asociada. Si bien estas arquitecturas podrían ofrecer mejoras cualitativas en tareas específicas, su uso debe ser cuidadosamente ponderado en función del contexto operativo y la infraestructura disponible.

Modelos como GSS y ST-MoE ofrecen tiempos intermedios, situándose en rangos entre 22 y 26 horas para ambos datasets. Esto los convierte en alternativas viables cuando se desea un equilibrio entre rendimiento y costo temporal, especialmente en escenarios donde no se justifica el despliegue de arquitecturas más pesadas como UT o MEGA.

6.2.2 Tiempo de validación

En una comparativa de soluciones también es esencial considerar el tiempo de validación, ya que este influye directamente en la eficiencia del ciclo de entrenamiento y evaluación. La validación se realiza periódicamente para monitorear el desempeño del modelo en datos no vistos y tomar decisiones sobre ajustes de hiperparámetros, puntos de parada o selección de modelos. Si una arquitectura requiere largos tiempos de validación, puede ralentizar significativamente el proceso de desarrollo y dificultar experimentaciones rápidas.

Además, en entornos donde se evalúan múltiples *checkpoints* o se aplica validación cruzada, estos costos se multiplican. Por tanto, medir el tiempo de validación permite valorar no solo la precisión del modelo, sino también su practicidad y eficiencia operativa en flujos de trabajo reales.

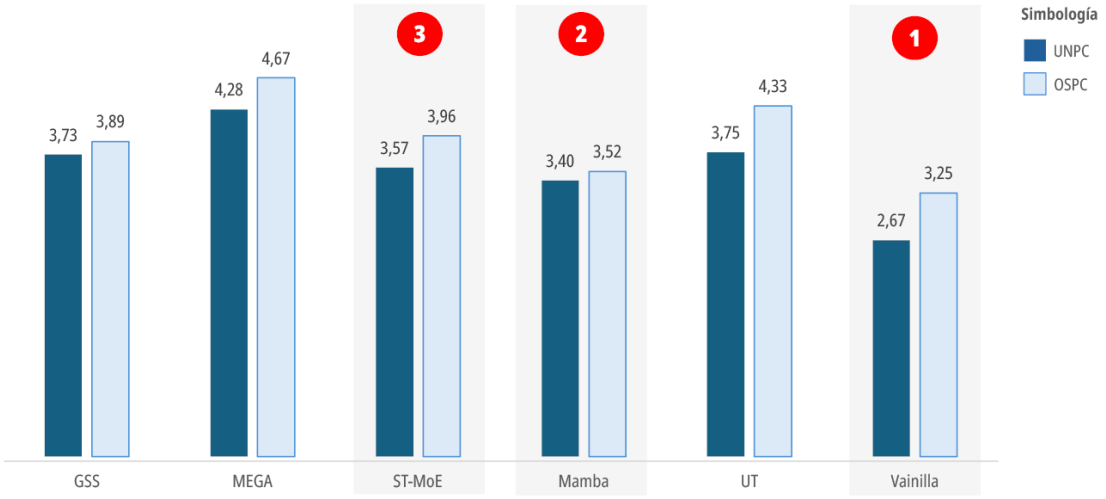
Cuadro No. 6.3. Tiempo total de la fase de validación de los modelos

Modelo	UNPC	OSPC
GSS	3,73	3,89
MEGA	4,28	4,67
ST-MoE	3,57	3,96
Mamba	3,40	3,52
UT	3,75	4,33
Vainilla	2,67	3,25

Nota: Tiempo de entrenamiento en horas.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.4. Tiempo total de la fase de validación de los modelos



Nota: Tiempo promedio dado en horas tras efectuar 20 epochs. Se utiliza UNPC para designar al conjunto de datos Corpus Paralelo de las Naciones Unidas (Ziemski et al., 2016) y OSPC para designar a OpenSubtitles v2018 (Lison & Tiedemann, 2016). Fuente: Elaboración propia del estudiante.

La figura anterior muestra los **tiempos totales de la fase de validación**, expresados en horas tras 20 épocas, para cada una de las arquitecturas evaluadas en los conjuntos de datos **UNPC** y **OSPC**. Esta métrica resulta clave, ya que la validación se ejecuta de forma periódica durante el entrenamiento para monitorear el desempeño del modelo en datos no vistos, impactando directamente en la agilidad del desarrollo y ajuste de modelos.

El tiempo requerido para completar los ciclos de validación es una métrica fundamental en la evaluación de arquitecturas de NMT, especialmente cuando se desea optimizar el proceso iterativo de mejora del modelo. En contextos de desarrollo ágil y experimentación continua, contar con ciclos de validación breves permite obtener retroalimentación más rápida sobre el desempeño del modelo, ajustar hiperparámetros con mayor eficiencia y acelerar los tiempos de entrega de soluciones entrenadas.

Como se observa, el modelo Vainilla nuevamente se posiciona como el más eficiente en esta métrica, registrando apenas 2,67 h con UNPC y 3,25 h con OSPC, confirmando su bajo costo computacional general tanto en entrenamiento como en validación.

El modelo Mamba ocupa el segundo lugar en eficiencia, con tiempos también muy contenidos: 3,40 h (UNPC) y 3,52 h (OSPC). Esta consistencia en rendimiento computacional, tanto en entrenamiento como en validación, refuerza su perfil como una arquitectura altamente eficiente y viable para despliegues en entornos con recursos limitados.

ST-MoE y GSS presentan tiempos de validación ligeramente superiores, aunque aún razonables: ambos se mantienen en el rango de las 3,5 a 4 horas. Sin embargo, es importante notar que MEGA y UT vuelven a posicionarse como las arquitecturas más demandantes, alcanzando hasta 4,67 h (MEGA con OSPC) y 4,33 h (UT con OSPC). Esta tendencia es consistente con sus tiempos de entrenamiento más extensos y sus estructuras más complejas, lo que implica mayores exigencias computacionales en todas las etapas del ciclo de vida del modelo.

Por tanto, el análisis de esta métrica confirma una vez más que las arquitecturas Vainilla y Mamba son las más eficientes desde la perspectiva computacional integral. Estas alternativas son ideales para escenarios donde la velocidad de iteración o los costos de hardware son factores críticos. Por otro lado, arquitecturas como UT y MEGA deben reservarse para casos donde el rendimiento superior justifique el sobrecosto computacional en entrenamiento y validación.

6.3 Consumo de memoria

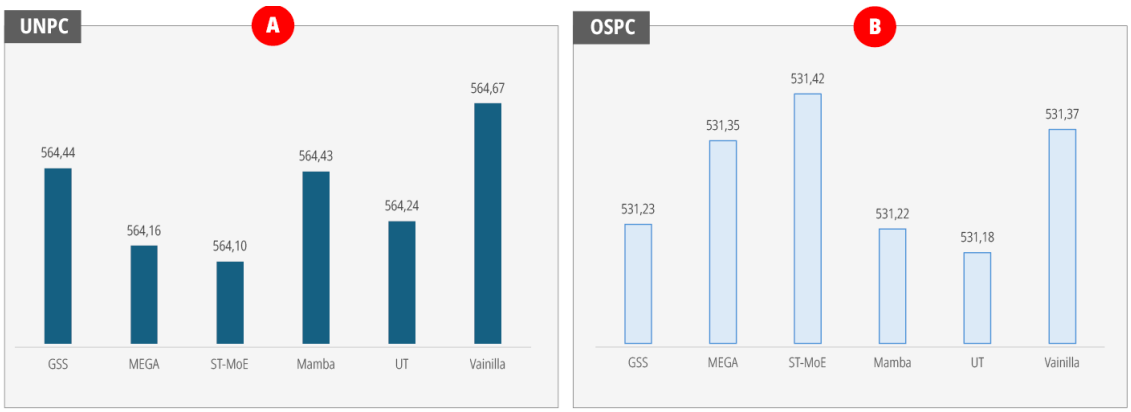
Medir la memoria que consume cada modelo en una comparativa de soluciones es crucial porque permite evaluar su viabilidad práctica en diferentes entornos de ejecución. Un modelo que consume excesiva memoria puede requerir hardware especializado o limitar el tamaño del lote (*batch size*), afectando directamente la velocidad de entrenamiento y la eficiencia durante la inferencia. Además, en escenarios donde se trabaja con secuencias largas o se desea escalar el modelo, el uso de memoria se convierte en un factor determinante para evitar errores por falta de recursos (*out-of-memory*) y optimizar el aprovechamiento del hardware disponible. Comparar el consumo de memoria también ayuda a identificar cuellos de botella y a seleccionar arquitecturas más adecuadas según las restricciones técnicas o económicas del proyecto, haciendo más realista y sostenible su implementación.

Cuadro No. 6.4. Total de memoria consumida por modelo

Modelo	UNPC	OSPC
GSS	564,44	531,23
MEGA	564,16	531,35
ST-MoE	564,10	531,42
Mamba	564,43	531,22
UT	564,24	531,18
Vainilla	564,67	531,37

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.5. Total de memoria consumida por modelo (en Megabytes)



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La información anterior sobre consumo de memoria presenta el promedio de uso en megabytes por lote (*batch*) tras 20 ejecuciones para los modelos evaluados en los conjuntos UNPC y OSPC. Esta métrica es fundamental, ya que el uso de memoria condiciona el tamaño del modelo que puede ser cargado en una unidad de procesamiento (CPU o GPU) y la cantidad de datos que se pueden procesar simultáneamente, afectando directamente la escalabilidad y la velocidad del entrenamiento.

En términos generales, no se observa una diferencia sustancial entre arquitecturas, ni tampoco entre corpus, lo cual sugiere que el factor memoria no es diferenciador en este conjunto específico de modelos y configuraciones. Esta conclusión es respaldada por la anotación explícita de la figura: *"No hay una marcada diferencia entre los diferentes modelos"*.

Para el conjunto UNPC, todos los modelos consumen entre 564,10 MB y 564,67 MB, siendo Vainilla el que registra el valor más alto y ST-MoE el más bajo, con una diferencia de apenas 0,57 MB. En el caso del conjunto OSPC, la variabilidad es incluso menor: los valores oscilan entre 531,18 MB y 531,42 MB, lo que indica una alta homogeneidad entre las arquitecturas analizadas en cuanto al uso de memoria.

Esta estabilidad en el consumo de memoria podría explicarse por la uniformidad en la configuración del entorno de entrenamiento (misma cantidad de capas, tamaño de batch y arquitectura base común), así como por la eficiencia del backend utilizado (PyTorch).

Desde una perspectiva aplicada, esto significa que la elección de una arquitectura más avanzada no conlleva necesariamente un mayor costo en términos de memoria, lo cual es una ventaja práctica para su implementación en infraestructuras con restricciones moderadas, como servidores de media capacidad o entornos de entrenamiento en la nube.

En conclusión, si bien métricas como el tiempo de entrenamiento o la cantidad de parámetros presentan variaciones importantes entre modelos, el consumo de memoria se mantiene prácticamente constante, lo que permite que la selección de arquitectura se base en criterios de rendimiento y eficiencia más que en limitaciones de hardware de memoria.

6.4 Pérdida

En una comparativa de soluciones para modelos de traducción automática, el análisis de la pérdida (*loss*) tanto en el conjunto de entrenamiento como en el de validación es crucial porque permite comprender cómo está aprendiendo el modelo y detectar problemas en el proceso de optimización que no siempre se evidencian con métricas de precisión o calidad final, como BLEU.

La pérdida mide el error cometido por el modelo al predecir la secuencia esperada, es decir, cuánto se desvía su salida con respecto a la traducción correcta. Una pérdida baja indica que el modelo está produciendo traducciones cercanas a las esperadas. Por tanto, analizar la pérdida durante el entrenamiento ayuda a evaluar si el modelo está aprendiendo progresivamente y si la optimización es estable y efectiva.

Por otro lado, la pérdida en el conjunto de validación refleja si ese aprendizaje se generaliza adecuadamente a datos no vistos. Un modelo con pérdida baja en entrenamiento pero alta en validación probablemente está sobreajustado: ha memorizado patrones específicos en lugar de aprender reglas generales del lenguaje. Inversamente, si la pérdida es alta en ambos conjuntos, puede haber un problema de subajuste (*underfitting*), indicando que la arquitectura o el entrenamiento no están capturando la complejidad del problema.

Comparar la pérdida en ambos conjuntos también permite evaluar la eficiencia del entrenamiento y anticipar posibles mejoras en hiperparámetros, regularización o arquitectura. Además, el análisis de la evolución de la pérdida entre épocas proporciona información temprana sobre la tasa de convergencia del modelo y su comportamiento en fases iniciales, lo cual es esencial para acelerar ciclos de desarrollo y evitar entrenamientos innecesariamente largos.

6.4.1 Fase de entrenamiento

A continuación se presentan los resultados del análisis de la **pérdida (loss)** obtenida por cada una de las arquitecturas evaluadas durante el proceso de entrenamiento en el conjunto de datos.

En este apartado, se exponen y comparan los valores iniciales y finales de pérdida para cada arquitectura, así como la variación porcentual entre ambos puntos, con el fin de identificar cuáles modelos muestran un aprendizaje más efectivo y sostenido a lo largo del proceso. Esta evaluación será fundamental para comprender en qué medida la complejidad de cada arquitectura se traduce en un mejor ajuste a los datos durante el entrenamiento.

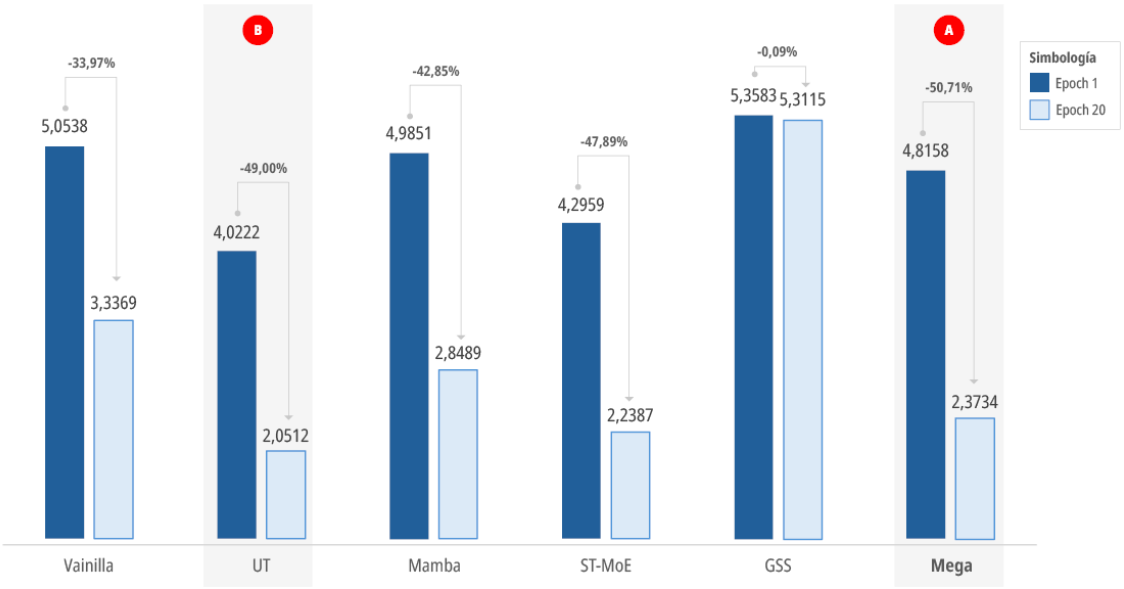
6.4.1.1 Conjunto de datos UNPC

Cuadro No. 6.5. Comparación de pérdida inicial y final durante la fase de entrenamiento en el conjunto de datos UNPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	5,0538	3,3369	-33,97%
UT	4,0222	2,0512	-49,00%
Mamba	4,9851	2,8489	-42,85%
ST-MoE	4,2959	2,2387	-47,89%
GSS	5,3583	5,3115	-0,87%
Mega	4,8158	2,3734	-50,72%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.6. Comparación de pérdida inicial y final durante la fase de entrenamiento en el conjunto de datos UNPC



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

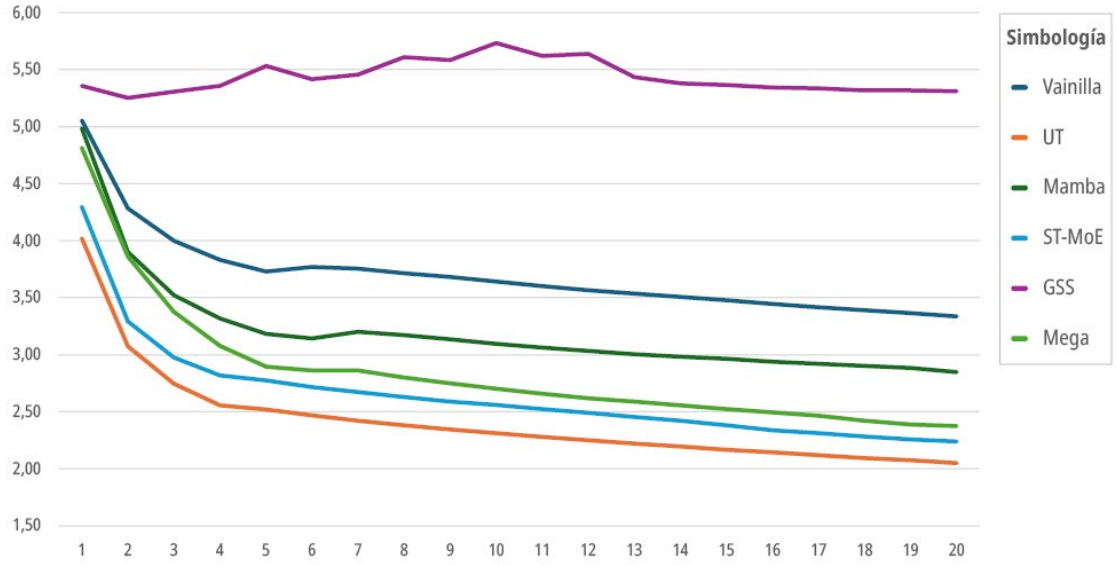
El **Cuadro No. 6.5 y el Gráfico No. 6.6** muestran la comparación entre la pérdida inicial (epoch 1) y final (epoch 20) de cada modelo entrenado sobre el conjunto de datos UNPC, lo que permite valorar la capacidad de aprendizaje de cada arquitectura y su eficiencia durante la fase de entrenamiento. La métrica de variación porcentual refleja la reducción relativa de la pérdida a lo largo del proceso.

El modelo MEGA destaca por lograr la mayor reducción de pérdida, con un -50,72 %, seguido de cerca por UT (-49,00 %) y MoE (-47,89 %). Estos resultados indican que dichas arquitecturas tienen un alto potencial de aprendizaje en contextos de lenguaje formal como el de UNPC. Mamba, aunque no alcanza el descenso más pronunciado, muestra una mejora sustancial con un -42,85 %, confirmando su solidez como arquitectura eficiente tanto en desempeño como en velocidad de convergencia.

Por otro lado, el modelo Vainilla obtiene una reducción del -33,97 %, lo que indica un desempeño aceptable, aunque inferior al de las arquitecturas más recientes. El caso más llamativo es GSS, que apenas mejora su pérdida inicial, con una variación de solo -0,87 %, lo que sugiere posibles problemas de ajuste o limitaciones en su capacidad para adaptarse a este corpus específico.

Este análisis refuerza la idea de que modelos como MEGA, UT, MoE y Mamba presentan una mejor capacidad de aprendizaje durante las primeras 20 épocas. No obstante, al combinar esta métrica con los resultados de eficiencia computacional previamente analizados, Mamba se posiciona como una opción especialmente equilibrada, ofreciendo una buena capacidad de reducción de pérdida con menor consumo de recursos en comparación con otros modelos más costosos computacionalmente.

Gráfico No. 6.7. Comportamiento de la pérdida de los modelos durante el entrenamiento para el conjunto de datos UNPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior muestra las curvas de pérdida durante el entrenamiento en el conjunto UNPC, permitiendo visualizar el comportamiento de cada arquitectura a lo largo de las 20 épocas analizadas. Esta representación es clave para entender la dinámica del aprendizaje y la eficiencia en la convergencia de los modelos evaluados.

Desde el inicio del entrenamiento, se observa que la mayoría de los modelos reducen su pérdida de forma progresiva, con diferencias claras en la velocidad de descenso y en el punto de estabilización. El modelo UT destaca por su curva más descendente y estable, alcanzando los valores de pérdida más bajos al finalizar el entrenamiento. Le siguen MEGA, MoE y Mamba, que también presentan un descenso sostenido y eficiente, lo que indica una capacidad efectiva de ajuste a los datos sin generar inestabilidad.

Por su parte, Vainilla muestra una reducción más lenta, con una curva más plana a partir de la mitad del entrenamiento, lo que sugiere una convergencia limitada. Sin embargo, el caso más crítico es el de GSS, cuya curva se mantiene elevada y presenta incluso oscilaciones leves a lo largo de las épocas, reflejando dificultades de aprendizaje o inestabilidad en el proceso de optimización.

En conjunto, esta visualización confirma que modelos como UT, Mamba y MEGA logran un descenso rápido y sostenido en la pérdida, lo que evidencia una mayor eficacia en el proceso de aprendizaje desde etapas tempranas. Este tipo de análisis resulta esencial para evaluar no solo el resultado final del entrenamiento, sino también la calidad y estabilidad del aprendizaje a lo largo del tiempo.

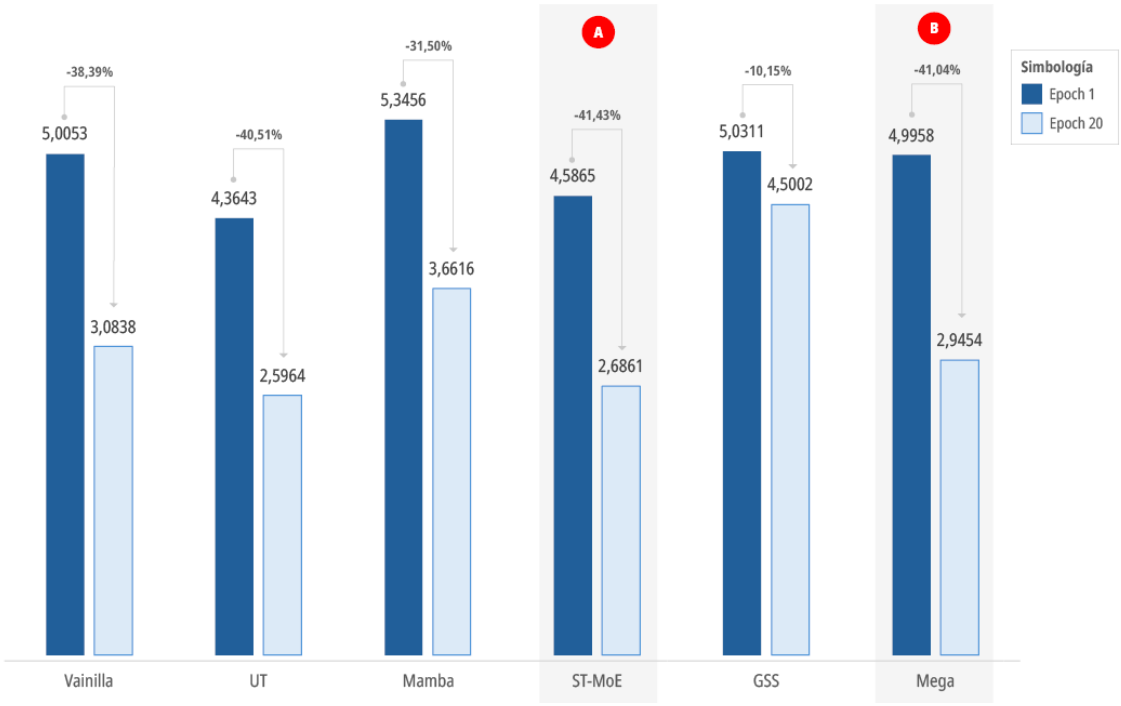
6.4.1.2 Conjunto de datos OSPC

Cuadro No. 6.6. Comparación de pérdida inicial y final durante la fase de entrenamiento en el conjunto de datos OSPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	5,0053	3,08380	-38,39%
UT	4,3643	2,59640	-40,51%
Mamba	5,3456	3,66160	-31,50%
ST-MoE	4,5865	2,68610	-41,43%
GSS	5,0311	4,50020	-10,55%
Mega	4,9958	2,94540	-41,04%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.8. Comparación de pérdida inicial y final durante la fase de entrenamiento en el conjunto de datos OSPC



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El cuadro y la gráfica anteriores presentan la comparación entre la pérdida inicial y final de cada modelo durante la fase de entrenamiento en el conjunto de datos OSPC, permitiendo evaluar la capacidad de ajuste de las arquitecturas en un entorno de lenguaje más informal y fragmentado. En este caso, se observa una menor magnitud de mejora generalizada en comparación con el corpus UNPC, lo que puede deberse a la mayor variabilidad y ruido presente en los datos.

Los resultados muestran que los modelos ST-MoE, UT y Mega alcanzan las reducciones más significativas de pérdida, con valores de -41,43 %, -40,51 % y -41,04 %, respectivamente. Esto indica que estas arquitecturas son especialmente efectivas para ajustarse a los datos del corpus OSPC durante el entrenamiento, al lograr un descenso notable en la función objetivo, lo que podría correlacionarse con una mejor capacidad de modelado de secuencias complejas.

El modelo Vainilla también presenta una mejora sustancial, con una disminución del -38,39 %, lo cual es destacable considerando su arquitectura más simple y su menor número de parámetros. En el caso de Mamba, la reducción es del -31,50 %, situándose en un nivel intermedio. Si bien su descenso porcentual es menor que el de otros modelos, esto debe ser interpretado en conjunto con

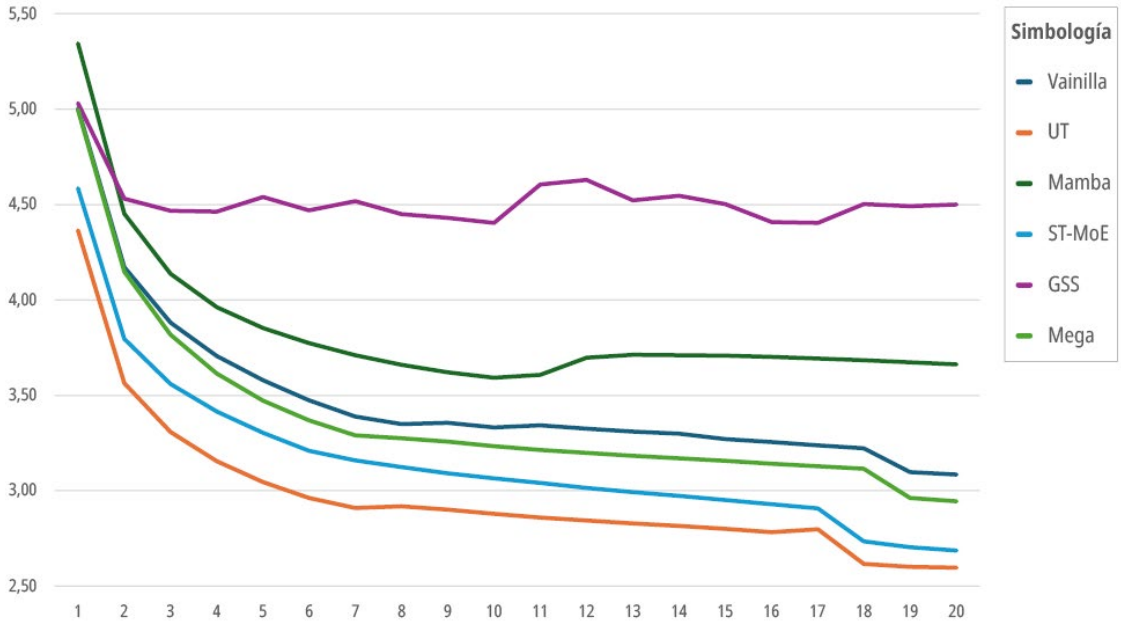
su eficiencia computacional superior observada en otras métricas (tiempo y memoria), lo que le da una ventaja costo-beneficio considerable.

Por el contrario, el modelo GSS muestra el comportamiento menos favorable, con una reducción de apenas -10,15 %, lo que confirma las observaciones anteriores sobre su curva plana de entrenamiento. Esta baja capacidad de ajuste puede deberse a una subutilización de la arquitectura, problemas de configuración o una menor compatibilidad con las características del corpus OSPC.

En términos absolutos, UT es el modelo que termina con la menor pérdida final (2,5964), seguido por ST-MoE y Mega, consolidando así su rendimiento superior desde el punto de vista del entrenamiento.

Este análisis cuantitativo respalda la conclusión de que los modelos más avanzados tienden a optimizar mejor la función de pérdida durante el entrenamiento, aunque con costos computacionales mayores. Modelos como Mamba y Vainilla ofrecen un equilibrio atractivo entre rendimiento y eficiencia, mientras que GSS requiere un replanteamiento en su implementación o parametrización para ser competitivo.

Gráfico No. 6.9. Comportamiento de la pérdida de los modelos durante el entrenamiento para el conjunto de datos OSPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica de curvas de pérdida durante el entrenamiento en el conjunto OSPC permite observar el comportamiento de aprendizaje de cada modelo en un entorno caracterizado por frases cortas, lenguaje informal y mayor variabilidad lingüística. En este contexto, la estabilidad y la capacidad de adaptación del modelo cobran especial relevancia, ya que los datos son menos estructurados que en el corpus UNPC.

Desde las primeras épocas, el modelo UT muestra una caída abrupta de la pérdida, lo cual indica una rápida capacidad de aprendizaje inicial. A partir de la época 10, su curva se estabiliza en torno a un valor cercano a 2,7, consolidándose como la arquitectura con la menor pérdida final de entrenamiento, lo cual podría reflejar una mejor capacidad de ajuste a los datos.

El modelo ST-MoE también exhibe un comportamiento favorable, con una convergencia relativamente rápida y una pérdida final ligeramente superior a la de UT. Le siguen Mamba y Vainilla, con descensos progresivos y estables, aunque con valores de pérdida final más elevados.

En contraste, el modelo GSS muestra una curva casi plana a lo largo del entrenamiento, sin una tendencia clara a la mejora. Esto sugiere una posible dificultad para aprender patrones representativos en este corpus específico o una configuración subóptima en sus hiperparámetros. Mega, aunque comienza con una pérdida alta, logra reducirla progresivamente y mantiene una trayectoria regular, aunque con una convergencia más lenta en comparación con UT y ST-MoE.

Este análisis permite concluir que UT y ST-MoE presentan las mejores dinámicas de aprendizaje durante el entrenamiento en OSPC, mientras que GSS evidencia limitaciones significativas en esta fase. Sin embargo, es importante considerar que una baja pérdida en entrenamiento no necesariamente garantiza una buena generalización, por lo que es indispensable complementar este análisis con las curvas de validación.

6.4.2 Pérdida durante la fase de validación

A continuación, se presentan los resultados del análisis de la pérdida (loss) durante la fase de validación para cada una de las arquitecturas evaluadas.

En esta sección se comparan los valores iniciales y finales de la pérdida en validación para cada arquitectura, así como la variación porcentual entre ambas mediciones. Este análisis permite determinar qué modelos logran un mejor equilibrio entre aprendizaje y generalización, y cuáles presentan dificultades para transferir el conocimiento adquirido durante el entrenamiento a

contextos nuevos. Esta información es esencial para seleccionar arquitecturas robustas y confiables en entornos de uso real.

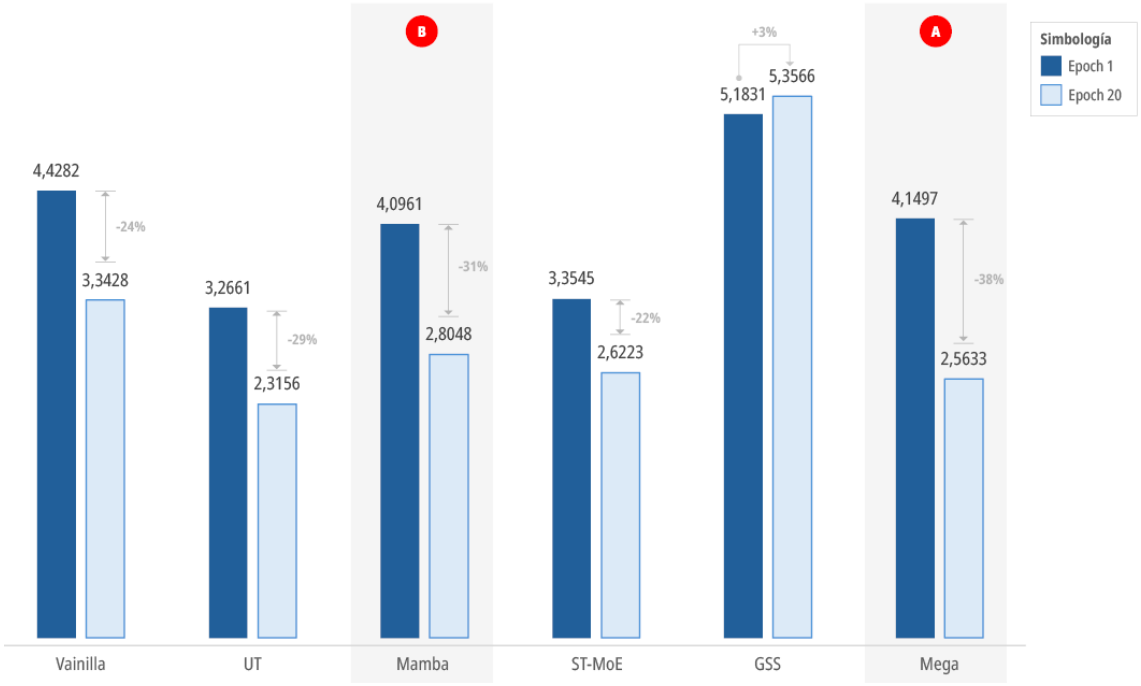
6.4.2.1 Conjunto de datos UNPC

Cuadro No. 6.7. Comparación de pérdida inicial y final durante la fase de validación en el conjunto de datos UNPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	4,4282	3,3428	-24,51%
UT	3,2661	2,3156	-29,10%
Mamba	4,0961	2,8048	-31,53%
ST-MoE	3,3545	2,6223	-21,83%
GSS	5,1831	5,3566	+3,35%
Mega	4,1497	2,5633	-38,23%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.10. Comparación de pérdida inicial y final durante la fase de validación en el conjunto de datos UNPC



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

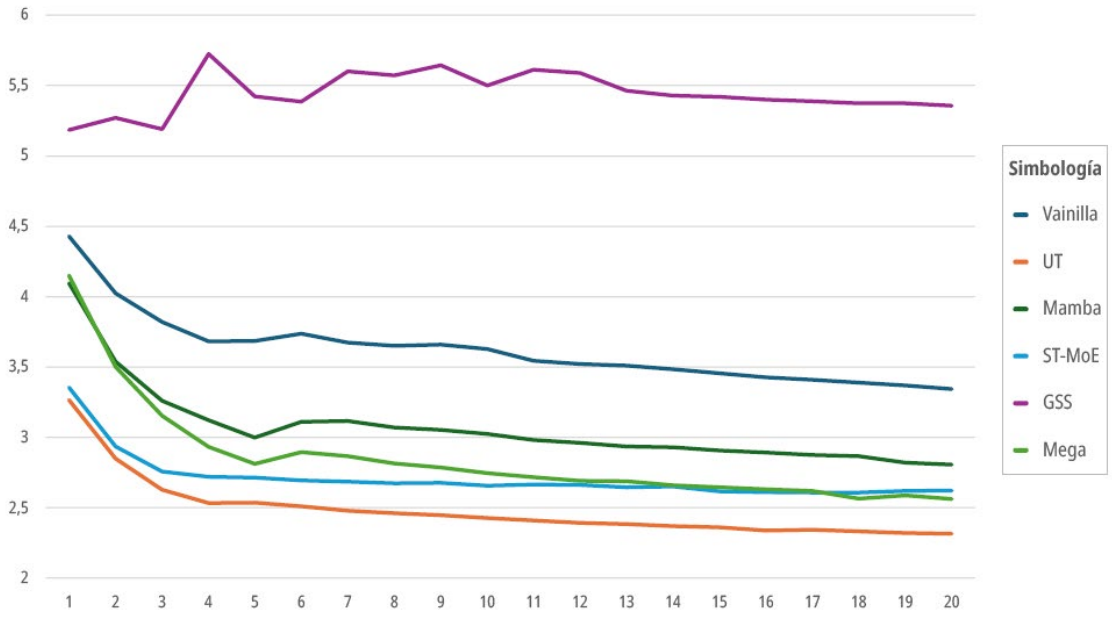
La información anterior muestra la evolución de la pérdida durante la fase de validación en el conjunto de datos UNPC, comparando los valores obtenidos en la primera y segunda época. Aunque el número de épocas analizadas es reducido, los resultados permiten observar la rapidez con la que cada modelo logra estabilizar su desempeño frente a datos no vistos, lo cual es indicativo de su capacidad de generalización temprana.

En este contexto, MEGA presenta la mayor mejora relativa con una reducción del -38,23 %, seguido de Mamba con -31,53 % y UT con -29,10 %. Estos resultados refuerzan la competitividad de estas tres arquitecturas en términos de rendimiento en validación desde las primeras etapas del entrenamiento. En particular, Mamba vuelve a demostrar una mejora sustancial, combinando buena capacidad de ajuste con un consumo moderado de recursos, lo que lo consolida como una opción robusta y eficiente.

El modelo Vainilla también logra una mejora destacable (-24,51 %), aunque inferior a la de los modelos más avanzados. En contraste, MoE presenta una mejora más modesta (-21,83 %), mientras que GSS es el único modelo que muestra un incremento en la pérdida de validación (+3,35 %), lo que sugiere sobreajuste prematuro o falta de generalización.

Estos datos, aunque limitados a dos épocas, aportan información relevante sobre la capacidad de respuesta temprana de cada arquitectura ante el proceso de validación. Una vez más, Mamba se perfila como una solución eficiente, que logra avances rápidos en precisión con bajo coste computacional, y con una buena capacidad de generalización desde las primeras fases del entrenamiento.

Gráfico No. 6.11. Comportamiento de la medida de pérdida de los modelos durante la validación para el conjunto UNPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica presentada muestra las curvas de pérdida durante la validación en el conjunto UNPC, permitiendo evaluar la evolución del error que cometen los modelos al enfrentarse a datos no vistos durante el entrenamiento. Esta medida es esencial para valorar la capacidad de generalización de cada arquitectura en un corpus estructurado y formal como el UNPC.

En términos generales, se observa que los modelos UT, MoE y MEGA mantienen las curvas más bajas y estables a lo largo de las 20 épocas, lo que indica un desempeño sólido y consistente ante datos nuevos. UT, en particular, destaca por lograr la menor pérdida en validación, con una trayectoria decreciente y sin oscilaciones relevantes, lo que evidencia un alto nivel de generalización y estabilidad. MoE y MEGA también reflejan curvas descendentes, aunque con variaciones leves, pero dentro de rangos aceptables.

Mamba, por su parte, presenta una curva ligeramente superior pero estable, con una pérdida controlada y una evolución progresiva. Aunque no es el modelo con menor pérdida absoluta, su comportamiento sugiere una buena capacidad de adaptación sin signos de sobreajuste, lo que refuerza su carácter equilibrado.

El modelo Vainilla muestra una pérdida claramente más elevada y una curva menos eficiente en su descenso, lo que evidencia una menor habilidad para generalizar. Finalmente, GSS presenta la peor evolución: su curva se mantiene alta, con fluctuaciones significativas y sin una reducción clara, lo que sugiere inestabilidad durante la validación y dificultades para aprender patrones generalizables del corpus.

En conjunto, esta visualización permite concluir que UT, MoE y MEGA logran las mejores tasas de generalización en UNPC, mientras que Mamba, sin ser el líder absoluto, ofrece un desempeño robusto con una arquitectura más eficiente. En cambio, Vainilla y GSS muestran limitaciones importantes que comprometen su aplicabilidad en tareas de validación sobre datos formales.

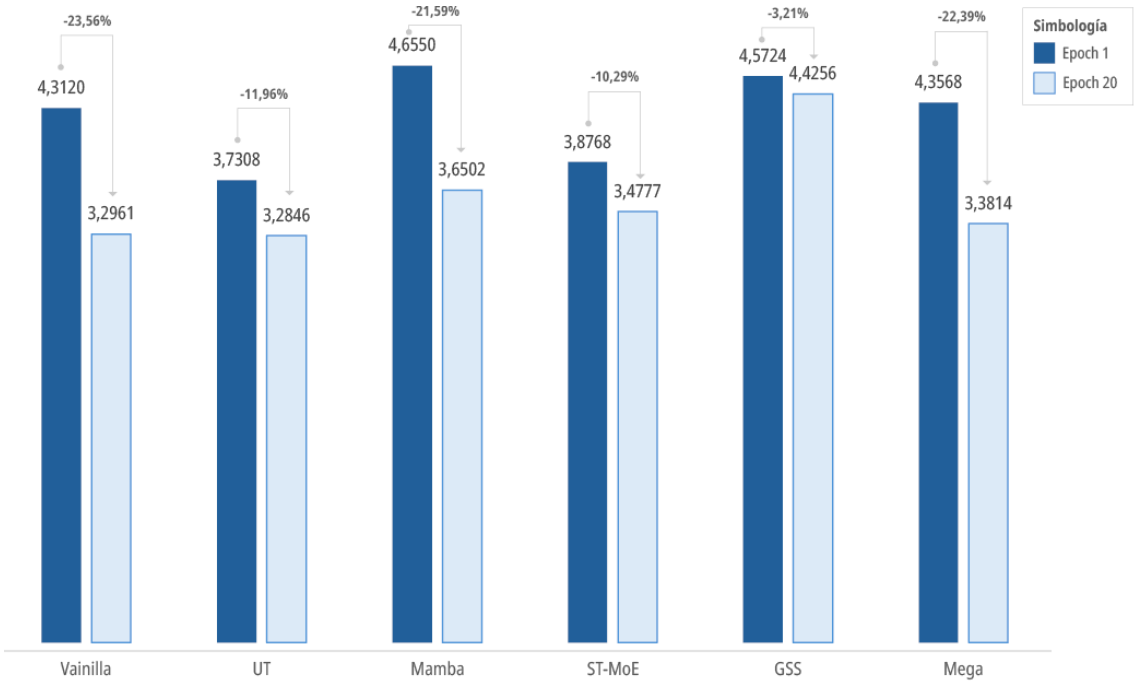
6.4.2.2 Conjunto de datos OSPC

Cuadro No. 6.8. Comparación de pérdida inicial y final durante la fase de validación en el conjunto de datos OSPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	4,3120	3,2961	-23,56%
UT	3,7308	3,2846	-11,96%
Mamba	4,6550	3,6502	-21,59%
ST-MoE	3,8768	3,4777	-10,29%
GSS	4,5724	4,4256	-3,21%
Mega	4,3568	3,3814	-22,39%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.12. Comparación de pérdida inicial y final durante la fase de validación en el conjunto de datos OSPC



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El **Cuadro No. 6.8** presenta los resultados de la pérdida de validación durante las primeras dos épocas de entrenamiento sobre el conjunto de datos OSPC, lo que permite evaluar la capacidad de generalización temprana de los modelos en un contexto de lenguaje informal y de frases cortas. A diferencia del conjunto UNPC, en OSPC se observan reducciones más discretas en la pérdida, probablemente debido a la naturaleza más ruidosa del corpus.

El comportamiento de la pérdida en la fase de validación es un indicador clave para evaluar la capacidad de generalización de un modelo de traducción automática. A diferencia de la pérdida en entrenamiento, que refleja cuán bien el modelo se ajusta a los datos vistos, la pérdida de validación permite detectar posibles problemas de sobreajuste o insuficiente aprendizaje con datos nuevos.

Entre los modelos evaluados, destacan por su mejor desempeño en validación:

- Vainilla, con una reducción del -23,56 %, mostrando que, a pesar de su arquitectura básica, es capaz de mejorar de forma sostenida y generalizar adecuadamente.
- Mega, con un descenso del -22,39 %, lo que respalda su competitividad en tareas prácticas más allá del entrenamiento.

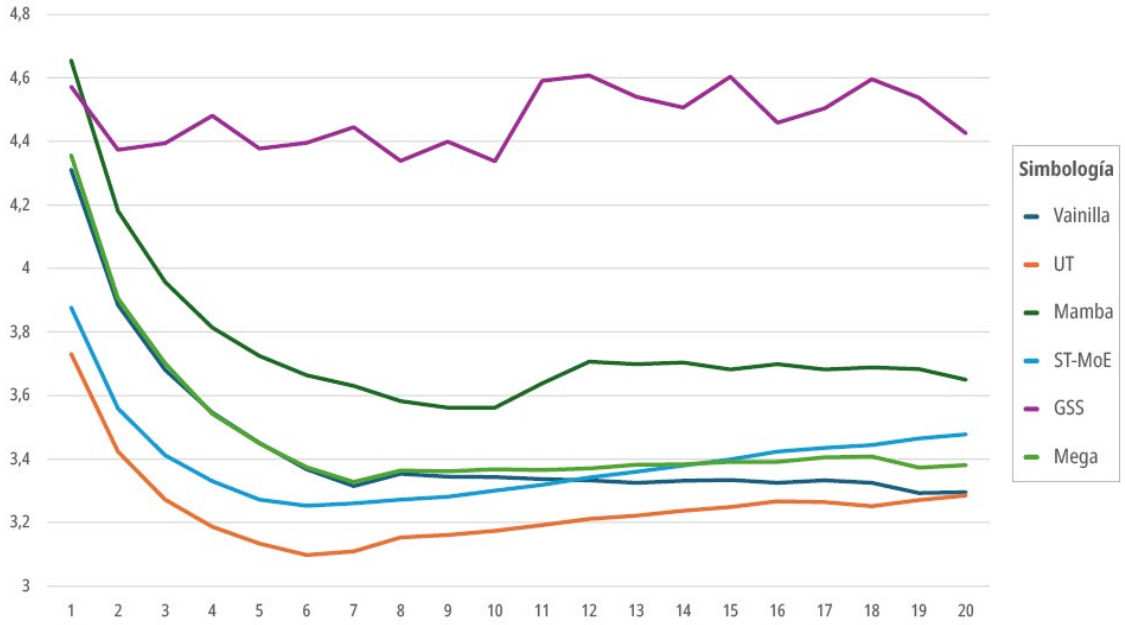
- Mamba, con una mejora del -21,59 %, confirma su solidez no solo como arquitectura eficiente, sino también robusta en términos de generalización.

El modelo UT, que había mostrado un excelente desempeño en pérdida de entrenamiento, presenta en este caso una mejora más modesta de -11,96 %, lo cual podría indicar un grado mayor de sobreajuste, es decir, un aprendizaje muy enfocado en los datos de entrenamiento con menor traspaso a los datos de validación.

Por su parte, ST-MoE presenta una mejora de -10,29 %, lo cual, si bien es moderado, refleja una mejora constante. Finalmente, GSS es nuevamente el modelo con el rendimiento más limitado en esta métrica, con tan solo -3,21 % de mejora, lo cual consolida su posición como la arquitectura con menor evolución durante el entrenamiento y validación en el corpus OSPC.

Este análisis confirma que la reducción de pérdida en validación no siempre guarda una correlación directa con la pérdida en entrenamiento. Modelos como Vainilla y Mamba destacan por su balance entre simplicidad, eficiencia computacional y capacidad de generalización, mientras que otros como UT, a pesar de su complejidad, requieren ajustes cuidadosos para evitar sobreajuste.

Gráfico No. 6.13. Comportamiento de la medida de pérdida de los modelos durante la validación para el conjunto de datos OSPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior presenta las curvas de pérdida durante la validación en el conjunto OSPC, ofreciendo una visión clara de cómo se comporta cada modelo al enfrentarse a un corpus informal, ruidoso y altamente variable como el de subtítulos. En este entorno, la capacidad de generalización y la estabilidad del aprendizaje son factores críticos para valorar la eficacia real de los modelos.

El análisis detallado de la evolución de la pérdida durante la validación permite observar no solo el rendimiento final de cada arquitectura, sino también su estabilidad y tendencia a generalizar adecuadamente. A través del monitoreo de la pérdida en cada época, es posible identificar patrones de sobreajuste, subajuste o aprendizaje sostenido. Se observan varios comportamientos diferenciados entre los modelos:

- El modelo UT muestra una rápida disminución de la pérdida durante las primeras siete épocas, alcanzando su valor mínimo alrededor de la época 7, seguido por una ligera tendencia ascendente. Este patrón sugiere una fase inicial de aprendizaje eficiente, seguida de un posible inicio de sobreajuste al final del entrenamiento.
- Vainilla, MoE y Mega presentan curvas estables y con descenso sostenido, aunque menos pronunciado. Esto indica un aprendizaje progresivo con buena estabilidad, lo que los convierte en opciones robustas cuando se busca generalización sin altos riesgos de sobreajuste.
- Mamba muestra una curva muy similar a la de Mega, con un mínimo claro entre las épocas 7 y 10, pero con una oscilación leve posterior, lo que podría deberse a fluctuaciones en los lotes de validación o a una sensibilidad moderada en la función de optimización.
- El modelo GSS evidencia el comportamiento más desfavorable: su pérdida se mantiene alta y con fluctuaciones significativas, sin una tendencia clara a la mejora. Esto refuerza lo ya identificado en los análisis anteriores, donde GSS mostró reducciones muy limitadas tanto en entrenamiento como en validación.
- Finalmente, ST-MoE mantiene una curva descendente con cierta oscilación, aunque logra estabilizarse en valores competitivos hacia el final del entrenamiento.

En síntesis, el análisis de las curvas confirma que UT logra la pérdida más baja en validación en etapas medias, pero tiende a estabilizarse o aumentar, mientras que Vainilla, Mega y Mamba muestran una mejor estabilidad general, haciendo de estos modelos opciones atractivas para tareas

donde se prioriza robustez y generalización. GSS, por el contrario, muestra un comportamiento errático que requiere ajustes importantes para ser viable.

6.5 Exactitud

En una comparativa de soluciones para modelos de traducción automática, el análisis de la precisión tanto en el conjunto de entrenamiento como en el de validación es fundamental para evaluar la calidad del aprendizaje del modelo y su capacidad de generalización.

La precisión en el conjunto de entrenamiento indica qué tan bien el modelo ha aprendido a reproducir los patrones lingüísticos presentes en los datos que ya ha visto. Sin embargo, una precisión alta en entrenamiento no garantiza un buen desempeño si el modelo ha memorizado los datos sin captar reglas generales del lenguaje, es decir, si ha incurrido en sobreajuste (*overfitting*).

Por eso, la precisión en el conjunto de validación (datos no vistos durante el entrenamiento) es clave para medir si el modelo generaliza adecuadamente a nuevos ejemplos. Una brecha amplia entre la precisión en entrenamiento y validación puede revelar problemas de sobreajuste, mientras que una precisión similar en ambos conjuntos suele indicar un modelo bien balanceado y más confiable para su uso en el mundo real.

Este análisis también permite comparar la estabilidad y consistencia de distintas arquitecturas: modelos que muestran un rendimiento alto en ambos conjuntos son preferibles, incluso si no son los más precisos en entrenamiento, ya que son más robustos ante variaciones del lenguaje y dominio. Por el contrario, un modelo que rinde muy bien solo en entrenamiento pero se degrada en validación, puede resultar poco fiable en tareas reales de traducción automática, donde se enfrentará constantemente a textos nuevos.

6.5.1 Conjunto de datos UNPC

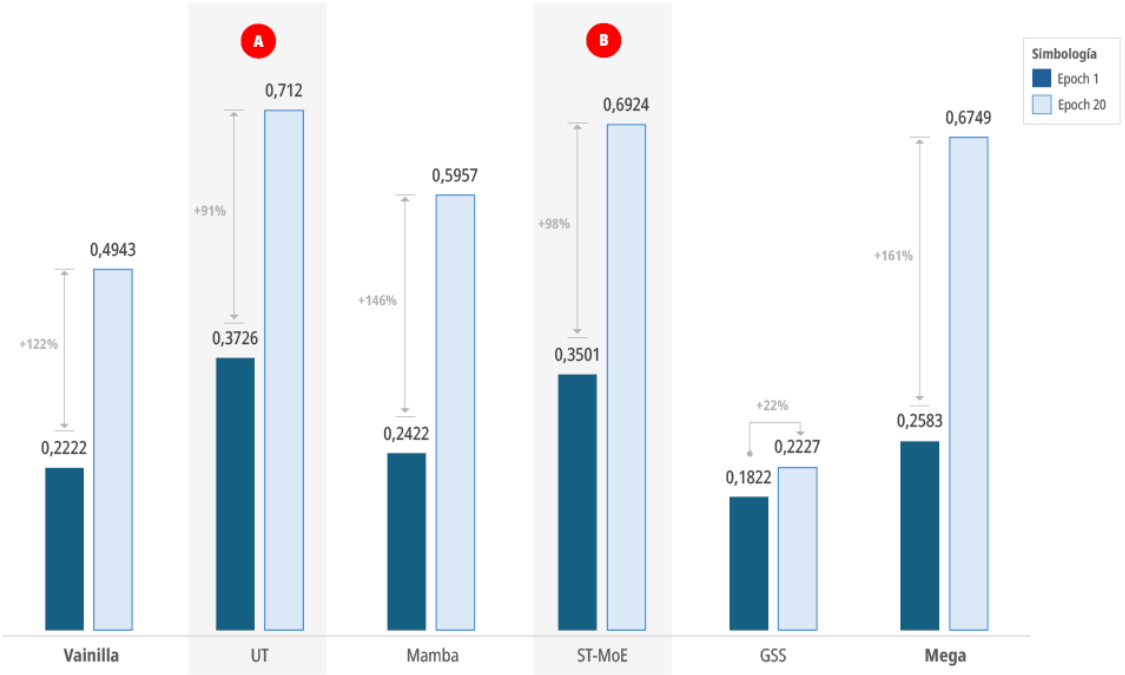
La tasa de mejora inicial es tan relevante como la precisión final, especialmente en contextos donde se requieren ciclos de entrenamiento cortos o donde se monitoriza el desempeño desde etapas tempranas.

Cuadro No. 6.9. Comparación de la exactitud durante la primera y última época de la fase de entrenamiento para el conjunto de datos UNPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	0,2222	0,4943	122,46%
UT	0,3726	0,7120	91,09%
Mamba	0,2422	0,5957	145,95%
ST-MoE	0,3501	0,6924	97,77%
GSS	0,1822	0,2227	22,23%
Mega	0,2583	0,6749	161,29%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.14 Comparación de la exactitud durante la primera y última época de la fase de entrenamiento para el conjunto de datos UNPC



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

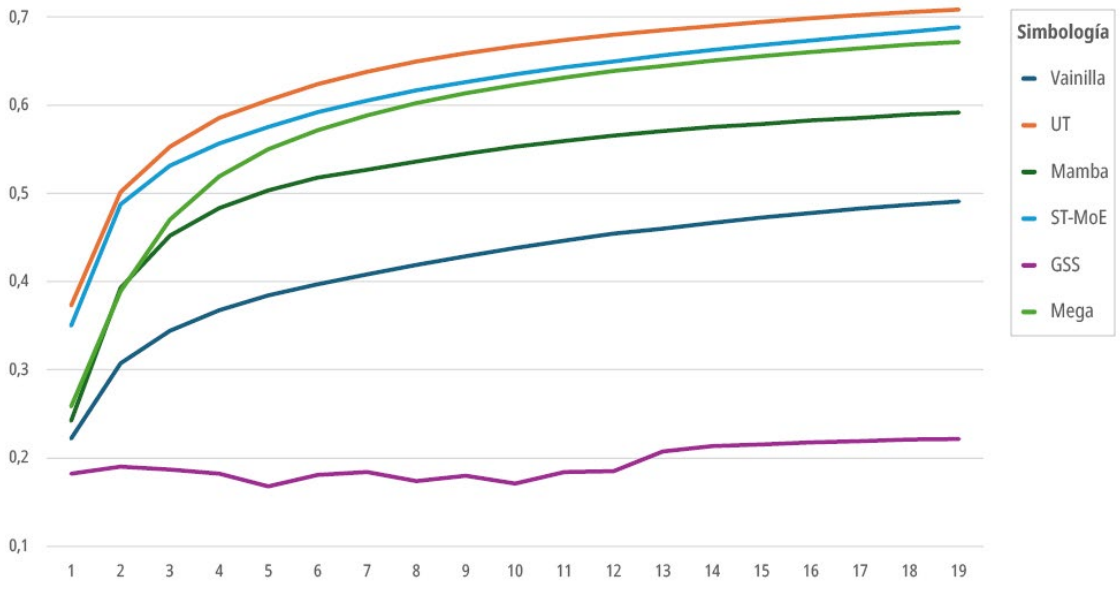
El **Cuadro No. 6.9** muestra la evolución de la precisión durante las dos primeras épocas de entrenamiento en el conjunto de datos UNPC, destacando la capacidad de los modelos para aprender de manera rápida en etapas iniciales. Esta métrica no solo refleja el ritmo de aprendizaje temprano, sino también la eficacia de la arquitectura para adaptarse al problema desde el inicio del entrenamiento.

Entre los modelos evaluados, MEGA y Mamba son los que presentan las mayores mejoras porcentuales, con un aumento del 161,29 % y 145,95 % respectivamente. Esto evidencia que ambas arquitecturas poseen una alta capacidad de aprendizaje desde las primeras iteraciones, lo que resulta ventajoso para entornos con limitaciones de tiempo o cuando se busca una convergencia rápida.

ST-MoE y UT también muestran mejoras significativas (97,77 % y 91,09 %), mientras que el modelo Vainilla mejora un 122,46 %, una cifra sorprendente considerando su simplicidad estructural, aunque partiendo desde una precisión inicial muy baja. El modelo GSS, en contraste, presenta una mejora marginal del 22,23 %, lo que refuerza su bajo rendimiento ya observado en métricas anteriores.

Estos resultados respaldan la eficacia temprana de modelos como Mamba y MEGA, siendo especialmente relevante en ciclos de entrenamiento reducidos, iteración rápida de prototipos o despliegues ágiles. En particular, Mamba logra este desempeño temprano con una arquitectura más ligera y tiempos de entrenamiento inferiores, consolidando su ventaja operativa frente a modelos más pesados.

Gráfico No. 6.15. Comportamiento de la medida de exactitud de los modelos durante el entrenamiento en el conjunto de datos UNPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior muestra las curvas de exactitud durante la fase de entrenamiento en el conjunto UNPC, permitiendo analizar cómo evoluciona la capacidad de los modelos para realizar

predicciones correctas a lo largo del tiempo. Esta métrica, entendida como la proporción de aciertos frente al total de ejemplos procesados, es útil para valorar la eficacia del aprendizaje supervisado en términos de rendimiento progresivo.

Desde las primeras épocas, se observa un crecimiento acelerado en los modelos UT, ST-MoE, Mamba y MEGA, todos los cuales superan rápidamente el umbral del 0,50 de exactitud. Entre ellos, UT lidera consistentemente con una curva ascendente y sostenida, alcanzando el nivel más alto de exactitud al finalizar el entrenamiento. ST-MoE y MEGA también logran resultados muy competitivos, aunque con una ligera desaceleración hacia el final. Mamba, por su parte, mantiene una progresión constante, situándose ligeramente por debajo de los anteriores pero con una trayectoria estable y sin caídas abruptas, lo que indica una buena capacidad de aprendizaje sin señales de sobreajuste temprano.

En contraste, el modelo Vainilla mejora a lo largo del tiempo pero con una pendiente más moderada, y se mantiene por debajo del 0,50 durante la mayor parte del proceso. Finalmente, el modelo GSS presenta un desempeño claramente inferior, con una curva plana y valores de exactitud significativamente más bajos que el resto, lo que sugiere una limitada capacidad de aprendizaje en este corpus.

En conjunto, esta visualización confirma que UT, ST-MoE y MEGA alcanzan los mayores niveles de exactitud durante el entrenamiento, pero también que Mamba logra un equilibrio entre buen rendimiento y estabilidad, lo cual lo posiciona como una opción confiable para tareas de traducción con estructuras lingüísticas formales como las del corpus UNPC.

Cuadro No. 6.10. Comparación de la precisión durante la primera y última época de la fase de validación para el conjunto de datos UNPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	0,2929	0,4971	69,72%
UT	0,4754	0,6774	42,49%
Mamba	0,3624	0,6128	69,09%
ST-MoE	0,4785	0,6489	35,61%
GSS	0,1978	0,2259	14,21%
Mega	0,3431	0,6520	90,03%

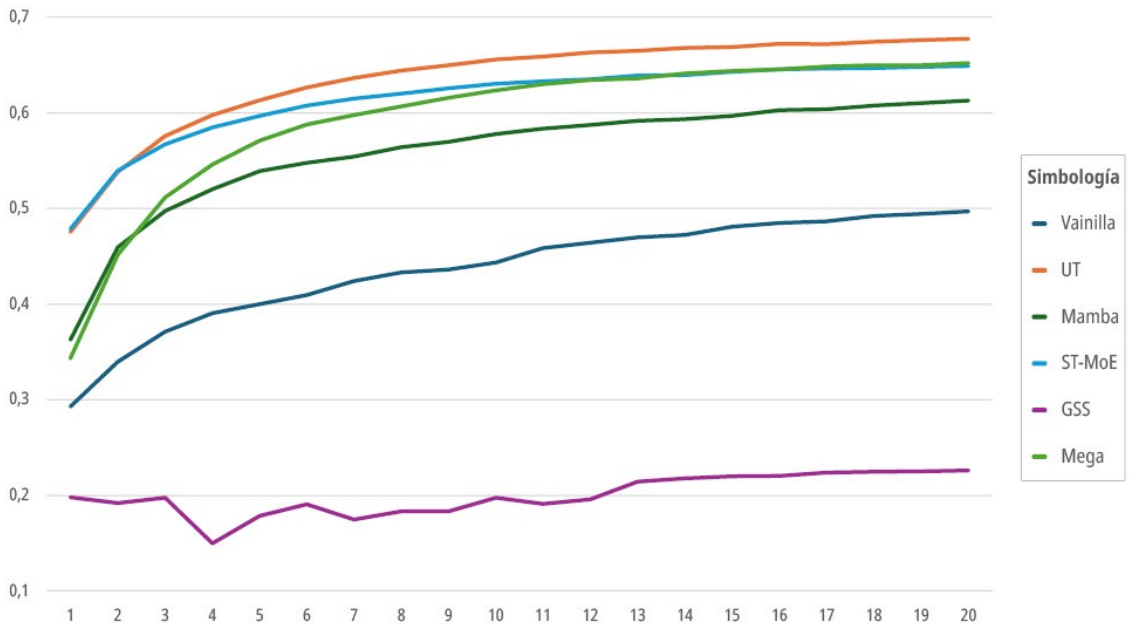
Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El cuadro anterior complementa el análisis anterior al mostrar la variación en precisión durante la fase de validación, entre la primera y la segunda época, en el conjunto de datos UNPC. Esta métrica permite observar no solo cuánto aprende el modelo, sino cómo se traslada ese aprendizaje a datos no vistos, lo cual es crucial para valorar la capacidad de generalización desde etapas tempranas.

El modelo con mayor mejora relativa en validación fue MEGA, con un incremento del 90,03 %, seguido por Vainilla (69,72 %) y Mamba (69,09 %). Esta mejora significativa confirma que Mamba, además de aprender con rapidez, generaliza eficazmente desde las primeras épocas, una característica especialmente valiosa en flujos de desarrollo donde el tiempo de entrenamiento está restringido o se aplican métodos de early stopping.

En contraste, modelos como UT (42,49 %) y MoE (35,61 %), aunque robustos en precisión final, muestran una menor velocidad de mejora en validación, lo que sugiere una curva de aprendizaje más gradual. GSS, con apenas un 14,21 % de mejora, vuelve a confirmar su debilidad en capacidad de generalización y escasa respuesta al entrenamiento en las primeras etapas.

Gráfico No. 6.16. Comportamiento de la medida de exactitud de los modelos durante la validación para el conjunto de datos UNPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior muestra las curvas de exactitud durante la fase de validación en el conjunto UNPC, permitiendo evaluar la capacidad de los modelos para generalizar el conocimiento adquirido durante el entrenamiento hacia datos no vistos, en un corpus estructurado y formal. La exactitud en validación es una métrica clave para identificar modelos robustos, capaces de mantener un rendimiento alto más allá del conjunto de entrenamiento.

El comportamiento general evidencia diferencias claras en la tasa y el techo de aprendizaje entre modelos:

- El modelo UT (línea naranja) destaca por alcanzar la exactitud más alta de todo el grupo, con una curva que se eleva rápidamente y se estabiliza en torno a 0,68, reflejando una excelente capacidad de generalización sostenida.
- ST-MoE (celeste) y Mega (verde oliva) siguen una tendencia similar, alcanzando valores cercanos a 0,65, aunque con una ligera desaceleración en las últimas épocas. Esto indica un aprendizaje eficiente sin indicios marcados de sobreajuste.
- El modelo Mamba (verde) muestra una curva ligeramente más baja pero estable, posicionándose en torno a 0,62 al final del entrenamiento. Su comportamiento gradual y consistente lo convierte en una opción robusta cuando se prioriza estabilidad sobre máximos teóricos de exactitud.
- El modelo Vainilla (azul oscuro), aunque inicia con una base razonable, presenta una curva más lenta y de menor alcance, culminando en torno a 0,50, lo que evidencia limitaciones en su capacidad de aprendizaje sobre el conjunto UNPC.
- Finalmente, el modelo GSS (línea morada) muestra una curva plana, con una exactitud inferior al 0,22, sin progresión notable a lo largo de las épocas. Este resultado refuerza las conclusiones previas respecto a su bajo rendimiento en todos los frentes evaluados (pérdida, tiempo, exactitud).

En conjunto, este análisis demuestra que las arquitecturas UT, ST-MoE y Mega logran un desempeño superior y sostenido en validación sobre UNPC, mientras que Mamba representa una opción balanceada entre rendimiento y eficiencia computacional. Por otro lado, Vainilla y especialmente GSS muestran limitaciones significativas que restringen su aplicabilidad práctica en corpus complejos.

Cuadro No. 6.11. Comparación de la precisión durante el entrenamiento y la validación en el conjunto de datos UNPC

Modelo	Entrenamiento	Validación
Vainilla	0,4943	0,4971
UT	0,7120	0,6774
Mamba	0,5957	0,6128
ST-MoE	0,6924	0,6489
GSS	0,2227	0,2259
Mega	0,6749	0,6520

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El cuadro anterior presenta los valores de precisión obtenidos durante las fases de entrenamiento y validación en el conjunto de datos UNPC, permitiendo valorar no solo el rendimiento de los modelos sobre los datos de entrenamiento, sino también su capacidad de generalización. Esta métrica es fundamental para determinar cuán bien el modelo está capturando patrones útiles sin sobreajustarse.

Los modelos que alcanzan mayores niveles de precisión tanto en entrenamiento como en validación son UT (0,7120 / 0,6774), MoE (0,6924 / 0,6489) y MEGA (0,6749 / 0,6520), lo cual confirma su buen rendimiento en términos de calidad de predicción. No obstante, como se ha analizado anteriormente, estas arquitecturas requieren más tiempo de entrenamiento y validación, mayor número de parámetros y, en algunos casos, mayor consumo de memoria.

En este contexto, Mamba se posiciona nuevamente como una opción intermedia robusta, logrando una precisión de 0,5957 en entrenamiento y 0,6128 en validación, cifras que si bien son más bajas que las de los modelos más pesados, superan con claridad al modelo Vainilla y al modelo GSS. Este resultado respalda la eficacia y consistencia de Mamba, especialmente al considerar su menor complejidad arquitectónica y sus ventajas en eficiencia computacional.

Por otro lado, el modelo GSS es el que presenta el desempeño más bajo en ambas métricas (0,2227 / 0,2259), lo que se alinea con su escasa reducción de pérdida y menor capacidad de aprendizaje observada en análisis anteriores.

Este cuadro complementa y reafirma las conclusiones anteriores: Mamba ofrece un equilibrio muy favorable entre precisión, eficiencia y simplicidad, lo que lo convierte en una de las arquitecturas más competitivas para tareas de traducción automática sobre corpus estructurados como UNPC.

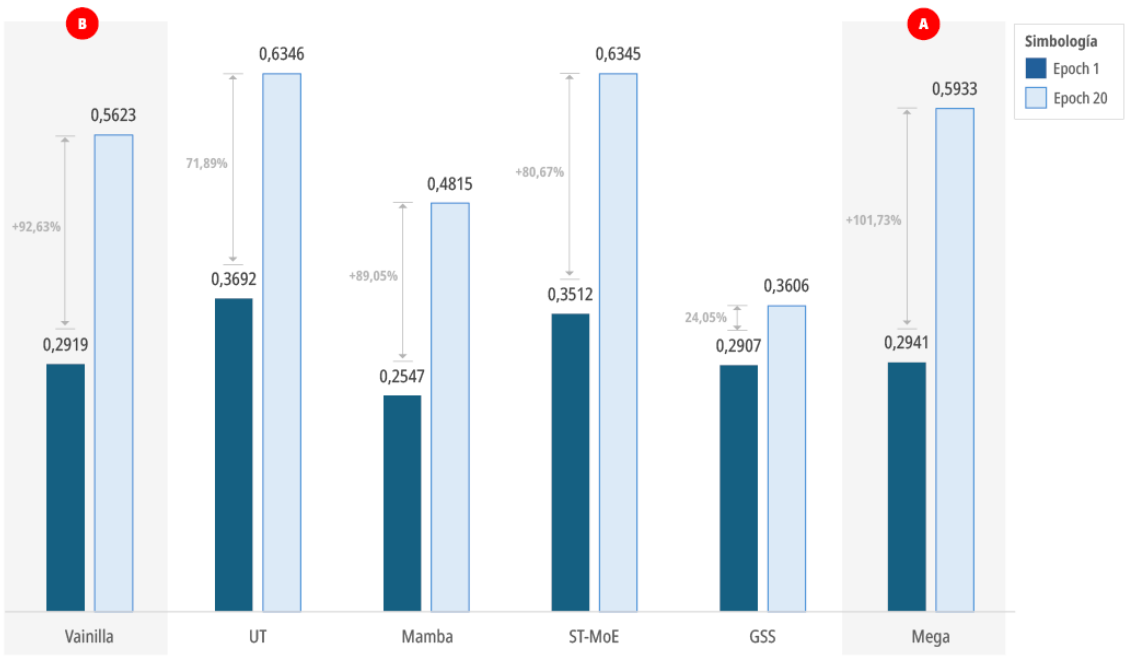
6.5.2 Conjunto de datos OSPC

Cuadro No. 6.12. Comparación de la exactitud durante la primera y última época de la fase de entrenamiento para el conjunto de datos OSPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	0,2919	0,5623	92,63%
UT	0,3692	0,6346	71,89%
Mamba	0,2547	0,4815	89,05%
ST-MoE	0,3512	0,6345	80,67%
GSS	0,2907	0,3606	24,05%
Mega	0,2941	0,5933	101,73%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.17. Comparación de la exactitud durante la primera y última época de la fase de entrenamiento para el conjunto de datos OSPC



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La información anterior presenta la variación en la precisión durante la fase de entrenamiento en el conjunto de datos OSPC, comparando la primera y segunda época. Este análisis permite evaluar la rapidez con la que cada modelo logra aprender en un entorno lingüístico más informal y fragmentado, lo que es fundamental para medir su capacidad de adaptación inicial. Se incluye además el porcentaje de mejora experimentado en ese intervalo.

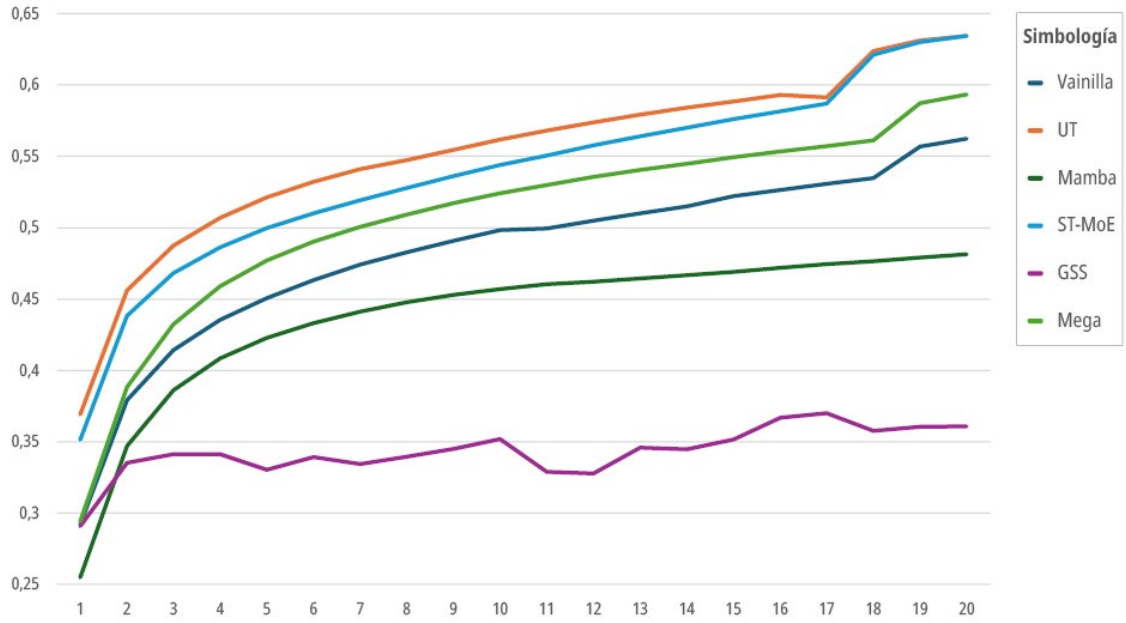
El modelo Mega destaca como el que mayor incremento de exactitud presenta, pasando de 0,2941 a 0,5933, lo que representa un aumento del +101,73 %, seguido muy de cerca por Vainilla (+92,63 %) y Mamba (+89,05 %). Esto evidencia que incluso modelos con arquitecturas relativamente simples (como Vainilla) pueden lograr grandes avances durante el entrenamiento, cuando están bien optimizados.

El modelo UT, que en otras métricas ha mostrado gran capacidad de aprendizaje, también logra un incremento importante, del +71,89 %, alcanzando una exactitud final de 0,6346, la más alta de todos los modelos evaluados. Esto confirma su capacidad de ajuste, aunque deberá contrastarse con las métricas de validación para valorar su generalización.

ST-MoE logra una exactitud final muy similar a UT (0,6345) con un crecimiento del +80,67 %, lo que confirma su eficiencia al combinar especialización con eficiencia de cómputo. Por otro lado, GSS vuelve a mostrar limitaciones, tanto en exactitud absoluta como en su incremento, alcanzando apenas un +24,05 % de mejora entre la primera y la última época.

En conclusión, este análisis confirma que la mayoría de modelos logran duplicar o casi duplicar su exactitud durante el entrenamiento, con UT, Mega y ST-MoE como líderes en desempeño final, mientras que GSS sigue rezagado, reforzando la evidencia de un bajo aprovechamiento de su arquitectura en el contexto del corpus OSPC.

Gráfico No. 6.18. Comportamiento de la medida de exactitud de los modelos durante el entrenamiento en el conjunto de datos OSPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior muestra el comportamiento de las curvas de exactitud durante el entrenamiento en el conjunto OSPC, permitiendo evaluar cómo evoluciona el rendimiento de cada modelo en un corpus caracterizado por su informalidad, menor estructura sintáctica y frases más breves. En este contexto, lograr una alta exactitud implica que el modelo ha podido adaptarse efectivamente a un entorno más variable y ruidoso.

Desde las primeras épocas, los modelos MoE y UT se destacan por su rápido ascenso en la medida de exactitud, manteniéndose como los modelos con mejor desempeño general al alcanzar valores cercanos al 0,65 al finalizar el entrenamiento. Su progreso es sostenido y continuo, lo que evidencia una curva de aprendizaje eficiente y una alta capacidad de adaptación a los patrones del corpus. Mamba y MEGA también muestran un crecimiento constante, aunque con curvas ligeramente más moderadas. Mamba, en particular, logra una evolución estable y sin caídas, consolidando su buen rendimiento pese a contar con una arquitectura más liviana.

El modelo GSS presenta una mejora gradual, aunque se mantiene en niveles inferiores en comparación con las demás arquitecturas. Sin embargo, su curva refleja una progresión más marcada que en el corpus UNPC, lo que podría indicar una mejor adaptación relativa en este contexto. Por el contrario, el modelo Vainilla muestra una curva plana y baja durante casi todo el entrenamiento, con

una evolución muy limitada en su exactitud, lo cual sugiere una dificultad para captar las particularidades lingüísticas del corpus OSPC.

En conjunto, esta visualización permite concluir que, en tareas de traducción con datos más informales, modelos como MoE, UT y Mamba no solo mantienen un buen rendimiento, sino que también evidencian una alta capacidad de aprendizaje y adaptabilidad, siendo adecuados para escenarios reales donde el lenguaje presenta mayor variabilidad.

Cuadro No. 6.13. Comparación de la precisión durante la primera y última época de la fase de validación para el conjunto de datos OSPC

Modelo	Epoch 1	Epoch 20	Variación
Vainilla	0,3627	0,541	49,16%
UT	0,4391	0,5454	24,21%
Mamba	0,3294	0,4935	49,82%
ST-MoE	0,4310	0,5353	24,20%
GSS	0,3351	0,3818	13,94%
Mega	0,3633	0,5415	49,05%

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El cuadro anterior muestra la evolución de la precisión durante la validación en el conjunto de datos OSPC, entre la primera y segunda época, lo que proporciona una visión sobre la capacidad de generalización temprana de los modelos en un entorno menos estructurado y más variable.

El seguimiento de la exactitud en validación permite estimar con precisión la capacidad de generalización de cada arquitectura ante datos no vistos. Mientras que la pérdida refleja el error, la exactitud en validación es un indicador directo de desempeño útil para tareas prácticas, especialmente en aplicaciones de traducción automática donde se requiere alta precisión en la salida.

Se observa que los modelos Mamba, Vainilla y Mega muestran los mayores incrementos relativos en exactitud, con mejoras del +49,82 %, +49,16 % y +49,05 %, respectivamente. Este comportamiento indica una curva de aprendizaje sostenida, junto con una buena capacidad para ajustar los parámetros sin comprometer la generalización.

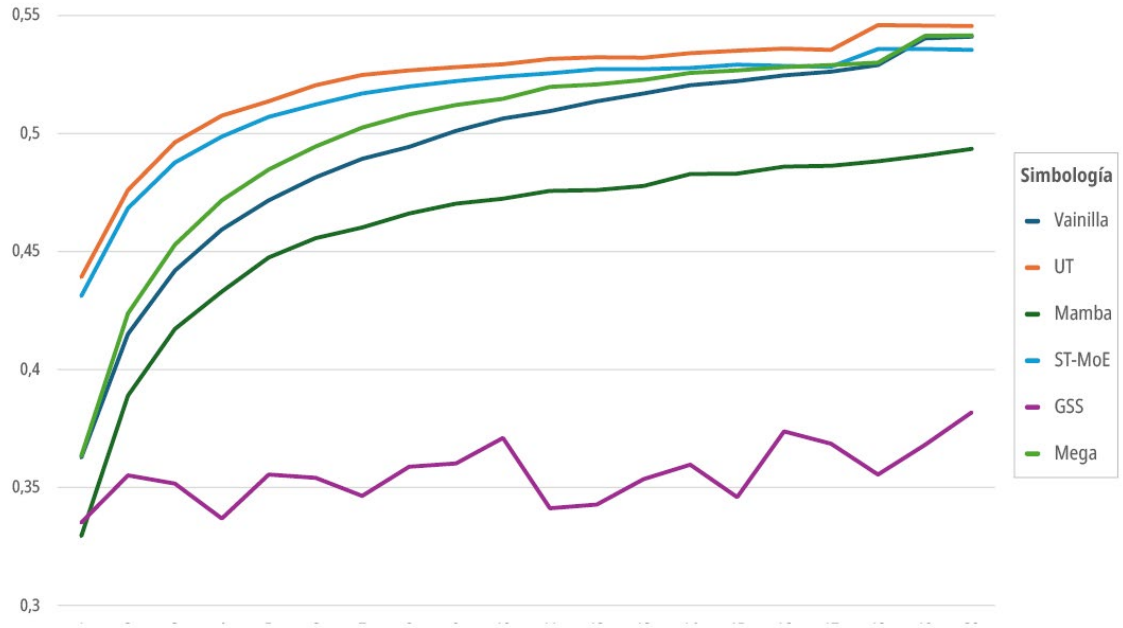
El modelo UT, si bien presenta la exactitud más alta al final de la validación (0,5454), muestra una mejora más contenida en términos relativos (+24,21 %), al igual que ST-MoE con +24,20 %. Esto

puede deberse a que ambos modelos parten de una base inicial más alta, lo que reduce el margen porcentual de mejora.

El modelo GSS vuelve a destacar negativamente con una mejora de solo +13,94 %, situándose consistentemente por debajo del resto en todos los indicadores de rendimiento revisados. Aunque supera ligeramente su punto de partida, su progreso es limitado, lo cual refuerza su bajo potencial de generalización en este contexto.

En resumen, este análisis confirma que, aunque UT y ST-MoE lideran en valores absolutos de exactitud, Mamba, Mega y Vainilla logran avances porcentuales sobresalientes, combinando simplicidad estructural con una capacidad efectiva de aprendizaje supervisado. Este resultado posiciona a estos modelos como alternativas altamente competitivas en entornos donde el equilibrio entre exactitud, eficiencia y recursos computacionales es crítico.

Gráfico No. 6.19. Comportamiento de la medida de exactitud de los modelos durante la validación para el conjunto de datos OSPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior muestra las curvas de exactitud durante la fase de validación en el conjunto OSPC, lo que permite analizar la capacidad de generalización de los modelos en un entorno más informal y menos estructurado. En este contexto, la evolución de la exactitud es un indicador clave

para identificar qué arquitecturas logran adaptarse con eficacia a la variabilidad del lenguaje. Mientras que las métricas puntuales (época 1 vs. 20) muestran el cambio entre dos momentos específicos, las curvas permiten visualizar la progresión del aprendizaje y detectar fenómenos como estancamiento, sobreajuste o crecimiento sostenido.

El modelo UT muestra la curva más alta a lo largo de todo el entrenamiento, alcanzando una exactitud superior a 0,54 al final del proceso. Esta tendencia evidencia un proceso de aprendizaje estable y efectivo, sin oscilaciones ni sobreajuste aparente.

ST-MoE y Mega mantienen trayectorias paralelas a UT, con un rendimiento final muy similar, apenas unas centésimas por debajo. Esto indica que ambas arquitecturas tienen un alto potencial de generalización, comparable al del modelo líder, pero posiblemente con menor costo computacional en ciertas configuraciones.

Mamba presenta una curva ligeramente inferior, aunque progresiva y constante, superando la barrera del 0,52 hacia la última época. Este comportamiento reafirma su perfil como un modelo balanceado, que mantiene buena estabilidad y eficiencia sin sacrificar rendimiento.

Vainilla muestra una mejora sostenida pero más moderada, finalizando en torno a 0,51. Aunque no alcanza los valores de las arquitecturas más avanzadas, su comportamiento confirma que sigue siendo una opción válida para tareas donde la simplicidad arquitectónica y la interpretabilidad sean prioritarias.

En contraposición, el modelo GSS muestra una curva baja y oscilante, que no supera el 0,38. Esta inestabilidad sugiere problemas de convergencia o inadecuación de su configuración para el corpus OSPC, lo cual ya había sido evidenciado en los resultados de pérdida y exactitud anteriores.

Este análisis confirma que UT, ST-MoE y Mega ofrecen los mejores desempeños de validación, con curvas altas, estables y sin señales de sobreajuste. Mamba y Vainilla se mantienen como alternativas confiables, mientras que GSS requiere una revisión profunda de arquitectura o parámetros.

Cuadro No. 6.14. Comparación de la precisión durante el entrenamiento y la validación en el conjunto de datos OSPC

Modelo	Entrenamiento	Validación
Vainilla	0,5623	0,5410
UT	0,6346	0,5454
Mamba	0,4815	0,4935
ST-MoE	0,6345	0,5353
GSS	0,3606	0,3818
Mega	0,5933	0,5415

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

De acuerdo con el cuadro anterior, estos son los resultados de precisión durante el entrenamiento y la validación en el conjunto de datos OSPC, completando así la comparativa del rendimiento de las distintas arquitecturas bajo condiciones de lenguaje informal y fragmentado. Esta métrica permite evaluar no solo qué tan bien el modelo aprende, sino también cuán efectivamente generaliza a nuevos datos en escenarios menos estructurados.

La evaluación conjunta de la precisión alcanzada durante las fases de entrenamiento y validación permite estimar el grado de generalización de los modelos propuestos. Esta comparación resulta particularmente útil para detectar fenómenos como el sobreajuste (overfitting), en los cuales el modelo logra una alta precisión en los datos de entrenamiento pero falla en mantener un rendimiento similar en datos no vistos. Por el contrario, una diferencia pequeña entre ambas métricas, o incluso una ligera mejora en validación, puede considerarse indicio de una adecuada regularización del modelo y una correcta capacidad de generalización.

El Cuadro No. 6.14 presenta los valores finales de precisión obtenidos por cada una de las seis arquitecturas evaluadas (Vainilla, UT, Mamba, ST-MoE, GSS y Mega) en el conjunto OSPC, al finalizar las 20 épocas de entrenamiento.

En primer lugar, el modelo Vainilla evidencia un comportamiento bastante equilibrado, con una precisión en entrenamiento de 0,5623 y una validación de 0,5410. Esta pequeña diferencia sugiere que, a pesar de ser la arquitectura más simple del conjunto, mantiene una buena coherencia entre fases, sin sobreajuste evidente.

El modelo UT (Universal Transformer), por el contrario, aunque alcanza la mayor precisión absoluta en entrenamiento (0,6346), reduce su rendimiento en validación a 0,5454. Esta diferencia relativamente amplia es indicativa de un posible sobreajuste, lo que podría comprometer su

desempeño en contextos donde la distribución de los datos de entrada difiera significativamente de los datos de entrenamiento.

De forma interesante, el modelo Mamba presenta una mejora en la fase de validación, pasando de 0,4815 en entrenamiento a 0,4935 en validación. Este comportamiento, poco habitual pero altamente deseable, sugiere que la arquitectura logra generalizar de forma más efectiva al enfrentarse a ejemplos no vistos. Este mismo fenómeno, aunque en menor medida, se repite en el caso de GSS, cuya precisión mejora de 0,3606 a 0,3818.

En cuanto a ST-MoE, se observa una dinámica similar a UT: alcanza un alto rendimiento en entrenamiento (0,6345), pero disminuye a 0,5353 en validación. Esta caída sugiere también una posible tendencia al sobreajuste, aunque su rendimiento absoluto sigue siendo competitivo.

Finalmente, el modelo Mega presenta una precisión en entrenamiento de 0,5933 y en validación de 0,5415, reflejando nuevamente una brecha considerable. Aunque sus valores globales son altos, esta diferencia indica que se debe monitorear cuidadosamente su comportamiento al enfrentarse a nuevos datos.

En síntesis, Mamba destaca por su capacidad de generalización, al presentar mejoras o una mínima diferencia entre entrenamiento y validación, lo que evidencia un buen balance entre aprendizaje y robustez. Por otro lado, las arquitecturas más complejas como UT, ST-MoE y Mega alcanzan los mayores niveles de precisión en entrenamiento, pero muestran signos de sobreajuste que deben considerarse al seleccionar modelos para entornos de producción o despliegue en datos con características no vistas previamente.

A continuación, se presenta el análisis del gap E–V (entrenamiento vs validación) a partir de la tabla actualizada. Esta métrica permite observar la diferencia entre el desempeño del modelo en los datos que ha visto (entrenamiento) y en los datos nuevos (validación). Un gap pequeño o positivo indica buena capacidad de generalización, mientras que un gap negativo más grande puede ser signo de sobreajuste.

Cuadro No. 6.15. Comparación de brechas entre la precisión durante el entrenamiento y la precisión durante la validación

Modelo	UNPC			OSPC		
	Entrenamiento	Validación	Gap E-V	Entrenamiento	Validación	Gap E-V
Vainilla	0,4943	0,4971	+0,0028	0,5623	0,5410	-0,0213
UT	0,7120	0,6774	-0,0346	0,6346	0,5454	-0,0892
Mamba	0,5957	0,6128	+0,0171	0,4815	0,4935	+0,0120
ST-MoE	0,6924	0,6489	-0,0435	0,6345	0,5353	-0,0992
GSS	0,2227	0,2259	+0,0032	0,3606	0,3818	+0,0212
Mega	0,6749	0,6520	-0,0229	0,5933	0,5415	-0,0518

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El Cuadro No. 6.15 presenta una comparación detallada de las diferencias entre la precisión alcanzada en entrenamiento y validación (denominada "Gap E-V") para seis arquitecturas evaluadas, considerando ambos conjuntos de datos: UNPC y OSPC. Esta brecha proporciona un indicador directo sobre el nivel de sobreajuste o generalización de los modelos. Una brecha negativa indica que el modelo rinde mejor en entrenamiento que en validación —lo que puede evidenciar sobreajuste—, mientras que una brecha positiva o cercana a cero sugiere mayor capacidad de generalización.

En el corpus UNPC, el modelo Vainilla destaca por tener una brecha prácticamente nula (+0,0028), lo que indica un rendimiento altamente consistente entre entrenamiento y validación. Este comportamiento reafirma la estabilidad de esta arquitectura en contextos menos exigentes o con menor complejidad sintáctica.

El modelo Mamba también presenta una brecha positiva (+0,0171), mostrando mayor precisión en validación que en entrenamiento, un comportamiento inusual pero favorable que ya se había observado en análisis anteriores. Algo similar ocurre con GSS (+0,0033), aunque en este caso, dada su baja precisión general, dicha mejora es menos relevante.

Por el contrario, los modelos ST-MoE y UT presentan brechas negativas más pronunciadas (-0,0435 y -0,0346, respectivamente), lo que sugiere la presencia de cierto grado de sobreajuste. Aunque ambos alcanzan valores absolutos de precisión altos, la diferencia indica que su ajuste al conjunto de entrenamiento puede no generalizarse plenamente a datos nuevos. Mega también presenta una brecha negativa (-0,0220), aunque más contenida.

Por su parte, El análisis sobre el corpus OSPC refuerza las observaciones anteriores. Nuevamente, Mamba y GSS muestran brechas positivas (+0,0120 y +0,0212, respectivamente), lo que indica una notable capacidad de generalización incluso en un corpus más complejo y diverso como OSPC.

Los modelos UT, ST-MoE y Mega vuelven a presentar las brechas negativas más amplias (-0,0892, -0,0992 y -0,0518, respectivamente), lo que confirma su tendencia al sobreajuste, especialmente cuando se utilizan en contextos de mayor variabilidad lingüística. A pesar de su alto desempeño en precisión durante el entrenamiento, esta diferencia sugiere que podrían no ser siempre la mejor opción si la generalización es el objetivo prioritario.

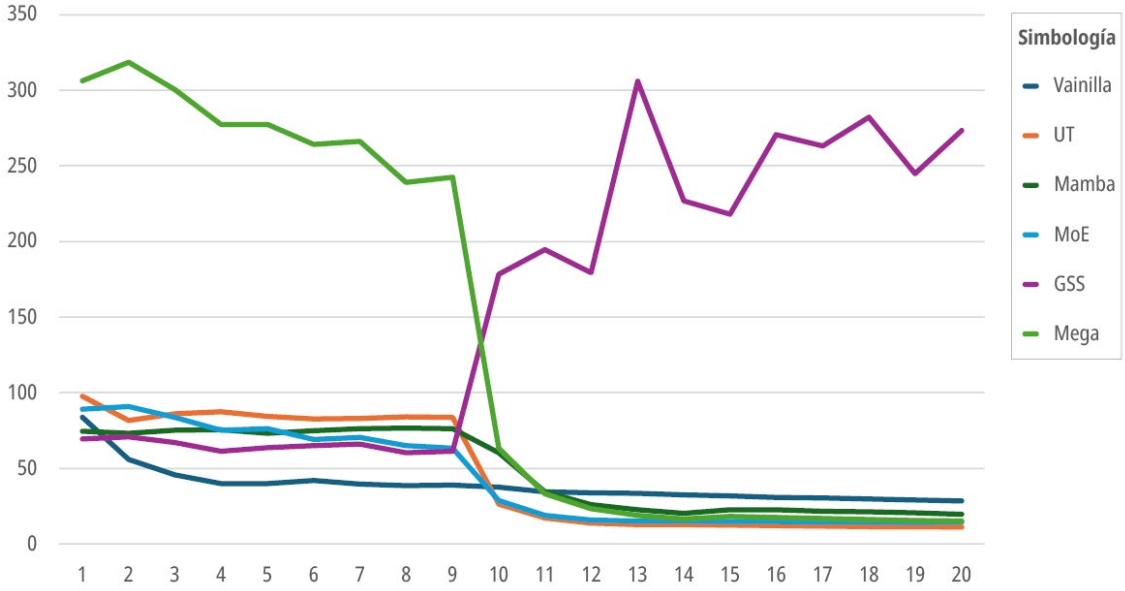
En cuanto al modelo Vainilla, su brecha negativa es pequeña (-0,0213), lo cual indica un rendimiento bastante consistente también en OSPC, aunque con menor precisión absoluta en comparación con modelos más avanzados.

La comparación entre conjuntos permite concluir que Mamba es el modelo con el mejor balance entre aprendizaje y generalización, ya que presenta brechas positivas o cercanas a cero en ambos corpus. GSS, aunque también muestra brechas positivas, debe ser evaluado con cautela por su bajo desempeño general. En contraposición, UT, ST-MoE y Mega, si bien son los modelos con mayor precisión durante el entrenamiento, muestran una pérdida notable de rendimiento en validación, lo cual sugiere una mayor propensión al sobreajuste. Finalmente, Vainilla se mantiene como un modelo estable y predecible, lo cual puede ser ventajoso en escenarios con limitaciones de recursos o necesidad de interpretabilidad.

6.5.3 Perplejidad

La diferencia entre la perplejidad de entrenamiento y validación es crucial para evaluar la capacidad de generalización del modelo y para identificar problemas como el sobreajuste o el subajuste.

Gráfico No. 6.20. Perplejidad durante la fase de validación en el conjunto de datos UNPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La gráfica anterior muestra el comportamiento de la perplejidad durante la validación en el conjunto UNPC, una métrica esencial para evaluar la capacidad de generalización de los modelos de lenguaje. La perplejidad mide qué tan bien predice un modelo una secuencia de palabras: cuanto menor el valor, mayor la certeza y fluidez de las predicciones. Una diferencia notable entre la perplejidad en entrenamiento y validación suele indicar problemas de sobreajuste o subajuste, por lo que esta curva es especialmente útil para identificar comportamientos anómalos durante la optimización.

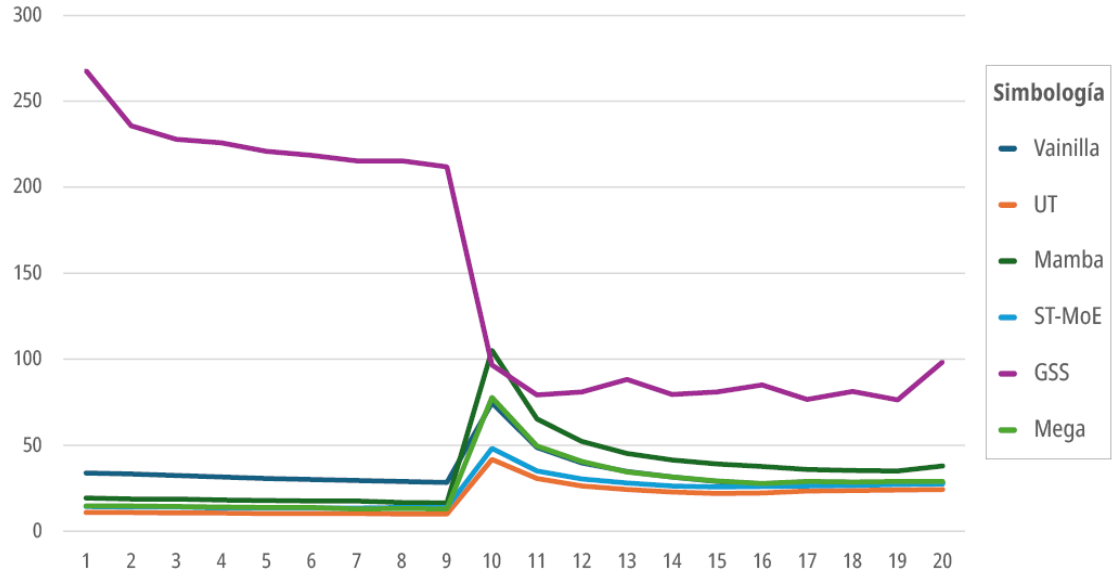
Los modelos UT, ST-MoE, Mamba y MEGA presentan los valores de perplejidad más bajos y estables hacia el final del entrenamiento, lo que indica un buen equilibrio entre aprendizaje y generalización. UT, en particular, logra estabilizarse por debajo de 30, con una curva descendente y sin oscilaciones, lo que refleja una alta consistencia en su desempeño. ST-MoE y Mamba también muestran trayectorias decrecientes con valores muy similares, mientras que MEGA, a pesar de un inicio más inestable, logra reducir abruptamente su perplejidad a partir de la época 10.

El modelo Vainilla mantiene una curva más elevada (alrededor de 40), lo que indica una menor capacidad de predicción en validación, aunque con un comportamiento relativamente estable. En cambio, GSS presenta una curva caótica y altamente fluctuante, con picos que superan los 300 en varias épocas, lo cual sugiere una fuerte inestabilidad y una deficiente capacidad de generalización.

Este comportamiento descalifica al modelo para tareas de traducción en contextos donde se requiere consistencia y precisión.

En conjunto, esta gráfica refuerza la evidencia de que Mamba, ST-MoE y UT son los modelos más robustos en el corpus UNPC, logrando bajos niveles de perplejidad y curvas estables, lo cual es clave para aplicaciones reales donde se requiere un modelo confiable, preciso y eficiente en entornos formales y estructurados.

Gráfico No. 6.21. Perplejidad durante la fase de validación en el conjunto de datos OSPC.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La mayoría de modelos presentan curvas regulares, con descensos progresivos o estables tras una fase inicial, a excepción del modelo GSS, que evidencia un comportamiento marcadamente anómalo.

Desde las primeras épocas, Vainilla, UT, Mamba, ST-MoE y Mega mantienen perplejidades relativamente bajas, con valores que oscilan entre 10 y 40 a partir de la época 10. En particular, UT y Mamba destacan por alcanzar los niveles de perplejidad más bajos y estables, lo que refleja una alta capacidad del modelo para generar secuencias coherentes con menor incertidumbre en la predicción. Estos modelos mantienen una tendencia descendente, lo cual indica que el proceso de entrenamiento logró mejorar sustancialmente la capacidad del modelo para ajustarse al patrón lingüístico del corpus.

ST-MoE y Mega, aunque presentan valores ligeramente más altos que UT, muestran una trayectoria igualmente estable y descendente, lo que sugiere buen aprendizaje sin signos de deterioro

o sobreajuste a lo largo del tiempo. El modelo Vainilla también mantiene una trayectoria aceptable, con una perplejidad intermedia y sin oscilaciones marcadas.

En contraste, el modelo GSS presenta un comportamiento altamente inestable y atípico. Durante las primeras 9 épocas, su perplejidad se mantiene por encima de 200, un valor que indica una fuerte incertidumbre en las predicciones del modelo. Aunque se observa una caída abrupta entre las épocas 9 y 10, posiblemente producto de una corrección en el aprendizaje o de la convergencia forzada del optimizador, la curva vuelve a subir y oscila significativamente en las épocas posteriores. Esta falta de estabilidad indica que el modelo no logra consolidar un aprendizaje consistente, lo cual está en línea con sus resultados desfavorables en otras métricas como pérdida y exactitud.

La perplejidad, como medida de la confianza en la generación de secuencias, confirma que los modelos UT, Mamba, ST-MoE y Mega son capaces de aprender de forma eficiente y generalizar adecuadamente en el corpus OSPC. Por otro lado, GSS vuelve a demostrar una inestabilidad grave, con perplejidades inaceptablemente altas durante una parte significativa del entrenamiento. El modelo Vainilla, aunque menos preciso en términos absolutos, mantiene un comportamiento predecible y razonable, lo que lo convierte en una base sólida para entornos controlados o de baja complejidad.

6.6 Inferencia

En una comparativa de soluciones para modelos de traducción automática, la evaluación con métricas como BLEU, el tiempo de inferencia promedio por secuencia y el tiempo total en los casos de prueba es esencial para obtener una visión completa del rendimiento de cada modelo. El BLEU (*Bilingual Evaluation Understudy*) es una métrica estándar que cuantifica la calidad de la traducción comparándola con traducciones humanas de referencia; permite evaluar cuán precisa y fluida es la salida del modelo, siendo clave para medir la fidelidad lingüística.

Por otro lado, el tiempo de inferencia promedio por secuencia indica la velocidad con la que el modelo traduce frases individuales, lo cual es crítico para aplicaciones en tiempo real o con requerimientos de baja latencia. Finalmente, el tiempo total en los casos de prueba refleja la eficiencia general del modelo al procesar grandes volúmenes de datos, ayudando a dimensionar su rendimiento en producción o en tareas de evaluación masiva. Considerar estas tres métricas de forma conjunta

permite balancear precisión y eficiencia, elementos clave en la elección de una arquitectura adecuada para sistemas de traducción automáticos.

6.6.1 BLEU

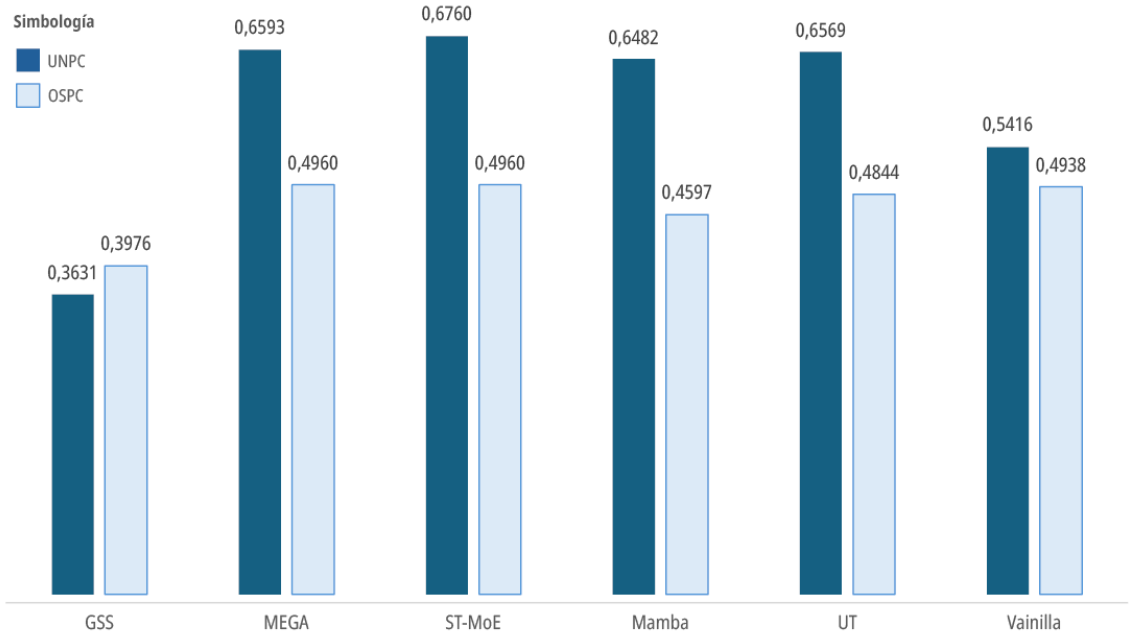
Como se mencionó en un inicio, en una comparativa de modelos de traducción automática, la métrica BLEU (*Bilingual Evaluation Understudy*) juega un papel fundamental como indicador objetivo de la calidad de las traducciones generadas por cada modelo. Su función principal es medir cuánto se parece la salida del modelo a una o más traducciones de referencia humanas, comparando n-gramas (secuencias de palabras) para evaluar precisión lingüística.

Cuadro No. 6.16. Resultados de la evaluación de la calidad de traducción mediante BLEU4.

Modelo	**Conjunto de datos**	
	UNPC	**OSPC**
GSS	0,3631	0,3976
MEGA	0,6593	0,4960
ST-MoE	0,6760	0,4960
Mamba	0,6482	0,4597
UT	0,6569	0,4844
Vainilla	0,5416	0,4938

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.22. Resultados de la evaluación de la calidad de traducción mediante BLEU4.



Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El cuadro y la gráfica anterior presentan los resultados de evaluación de calidad de traducción mediante la métrica BLEU, tanto en el conjunto de datos UNPC como en OSPC, lo que permite comparar de forma directa la fidelidad de las traducciones generadas por cada modelo con respecto a una referencia humana. BLEU es una métrica ampliamente utilizada en traducción automática, ya que captura la coincidencia de n-gramas entre la predicción del modelo y la traducción de referencia.

La evaluación de inferencia permite medir qué tan cercanas son las traducciones automáticas generadas por los modelos respecto a las traducciones de referencia realizadas por humanos. Para ello, se emplea la métrica BLEU-4, ampliamente utilizada en traducción automática, la cual se basa en el grado de coincidencia n-gram entre la salida del modelo y las oraciones de referencia. Un valor alto en BLEU indica una mayor calidad de traducción.

En general, todos los modelos muestran un desempeño superior en UNPC, lo cual es esperable dada la mayor regularidad y estructura del corpus (formado por frases más formales y menos ambiguas). Sin embargo, la variación entre corpus permite observar qué modelos mantienen mejor su capacidad de traducción al enfrentarse a datos más informales y desafiantes como los del conjunto OSPC.

Los modelos que obtienen los mejores valores absolutos en BLEU-4 en el conjunto UNPC son ST-MoE (0,6760), UT (0,6569), MEGA (0,6593) y Mamba (0,6482).

Estos resultados consolidan a ST-MoE y UT como arquitecturas particularmente eficaces en tareas de inferencia, capaces de producir traducciones muy próximas a las humanas en contextos estructurados.

Al evaluar el rendimiento sobre OSPC, todos los modelos sufren una caída, pero algunos logran conservar valores relativamente altos. En particular, UT mantiene un valor de 0,4844, ST-MoE y MEGA empatan con 0,4960 y Vainilla alcanza 0,4938, superando incluso a Mamba en este corpus.

Esto sugiere que, aunque Vainilla y Mega no alcanzan los picos más altos en entrenamiento, su capacidad de generalización en inferencia es competitiva. ST-MoE, por su parte, destaca por su consistencia entre ambos corpus, siendo el modelo con la menor caída relativa de rendimiento.

El modelo GSS, aunque mejora ligeramente de UNPC (0,3631) a OSPC (0,3976), se mantiene en el nivel más bajo de desempeño en inferencia, en línea con los hallazgos anteriores en pérdida, exactitud y perplejidad.

La métrica BLEU-4 confirma que los modelos ST-MoE, UT y Mega ofrecen las mejores traducciones automáticas desde el punto de vista de similitud con las referencias humanas. Mamba y Vainilla muestran un rendimiento robusto y sorprendentemente competitivo en OSPC, lo que los posiciona como alternativas viables para aplicaciones en lenguaje más informal o abierto. Por otro lado, GSS continúa mostrando limitaciones sustanciales, con valores bajos de precisión tanto en tareas de clasificación como en generación.

6.6.2 Tiempo de inferencia

En una comparativa de algoritmos de traducción automática neuronal, uno de los factores más importantes a tener en cuenta es el tiempo de inferencia, es decir, el tiempo que tarda un modelo en procesar una entrada y generar una salida.

La importancia de comparar el tiempo de inferencia radica en varios aspectos fundamentales que pueden influir en la elección del modelo más adecuado para una aplicación específica. A continuación se detallan las razones más importantes para realizar esta comparación.

En aplicaciones donde la velocidad es crítica, como sistemas de traducción en tiempo real (chats, servicios de atención al cliente, subtitulación automática o interpretación), el tiempo de inferencia es un factor decisivo.

Si el tiempo de inferencia es elevado, el sistema podría introducir retrasos notables (latencia), lo que afectaría la experiencia del usuario. En este caso, un modelo más rápido, aunque ligeramente menos preciso, podría ser preferido.

También, un tiempo de respuesta rápido es esencial para aplicaciones interactivas en las que los usuarios esperan traducciones instantáneas.

Para servicios que manejan grandes volúmenes de peticiones de traducción, como aplicaciones globales o servicios en la nube que gestionan millones de usuarios, la eficiencia del modelo en cuanto a tiempo de inferencia tiene un impacto directo en la capacidad de la plataforma para escalar. Un modelo con un tiempo de inferencia elevado puede aumentar los costos operativos, ya que requerirá más recursos computacionales para manejar el mismo número de solicitudes. Modelos más rápidos pueden traducir más peticiones en el mismo período, lo que permite un uso más eficiente de la infraestructura y reduce la necesidad de ampliar servidores.

Los modelos de traducción automática más complejos, como los basados en arquitecturas profundas (transformers con muchos parámetros), tienden a ser más precisos pero también más lentos. Por tanto, la elección del modelo dependerá de la prioridad de la aplicación:

- Aplicaciones sensibles a la precisión: En casos donde la calidad de la traducción es primordial, como traducciones legales o médicas, podría ser aceptable un tiempo de inferencia más largo si garantiza una mayor precisión.
- Aplicaciones donde la velocidad es crítica: En otras aplicaciones, como traducción de redes sociales o contenido en vivo, se puede optar por un modelo menos preciso pero con menor latencia para garantizar una experiencia más fluida.

Por otro lado, en ciertos escenarios, como la implementación en dispositivos con recursos limitados (smartphones, tabletas o dispositivos IoT), el tiempo de inferencia de un modelo puede ser limitado por las capacidades de procesamiento del dispositivo. Modelos más ligeros y optimizados pueden ser necesarios para operar eficientemente en dispositivos móviles, donde la capacidad de

procesamiento y el consumo energético son factores clave. Comparar el tiempo de inferencia en estos contextos permite elegir un modelo que funcione dentro de estas restricciones.

En cualquier servicio o aplicación orientada al consumidor, la experiencia del usuario es un factor clave. Los usuarios tienden a preferir aplicaciones que ofrezcan resultados rápidos y efectivos. Un sistema de traducción que ofrezca traducciones en milisegundos es más probable que sea adoptado que uno que tarde varios segundos, incluso si la diferencia en precisión es marginal.

La mayoría de los usuarios tienen una tolerancia limitada a los retrasos, especialmente en aplicaciones en línea. Un tiempo de inferencia rápido mejora significativamente la satisfacción del usuario.

En entornos donde el consumo de energía es importante, como centros de datos o dispositivos móviles, un modelo que ofrezca un tiempo de inferencia más corto también podría implicar un menor consumo de energía, lo cual es beneficioso tanto en términos de costos operativos como de sostenibilidad ambiental.

6.6.2.1 Tiempo promedio de inferencia

En una comparativa de soluciones para modelos de traducción automática, el análisis del tiempo promedio de inferencia por registro (o *query*) resulta especialmente relevante porque permite evaluar la eficiencia operativa del modelo en condiciones reales de uso. A diferencia de métricas como la precisión o el BLEU, que miden la calidad de la traducción, el tiempo de inferencia refleja cuán rápido es el modelo al generar una traducción para una entrada concreta, lo cual es crucial en múltiples escenarios prácticos.

En primer lugar, en aplicaciones de traducción en tiempo real, como asistentes de voz, subtitulación automática o sistemas de mensajería multilingüe, la latencia por *query* determina directamente la experiencia del usuario. Un modelo con alta calidad pero con tiempos de respuesta lentos puede resultar inviable en contextos donde la fluidez y la inmediatez son prioritarias.

En segundo lugar, esta métrica permite comparar modelos en términos de costo computacional por unidad procesada, lo que es fundamental en sistemas desplegados en la nube, aplicaciones móviles o entornos con recursos limitados. Un modelo con menor tiempo de inferencia puede procesar más solicitudes por segundo, optimizando el uso de hardware y reduciendo los costos operativos.

Además, el tiempo promedio de inferencia por registro sirve como indicador de escalabilidad, permitiendo prever cómo se comportará el sistema ante volúmenes crecientes de solicitudes. En sistemas de traducción masiva o procesos por lotes, incluso pequeñas diferencias por *query* pueden representar horas de procesamiento acumulado.

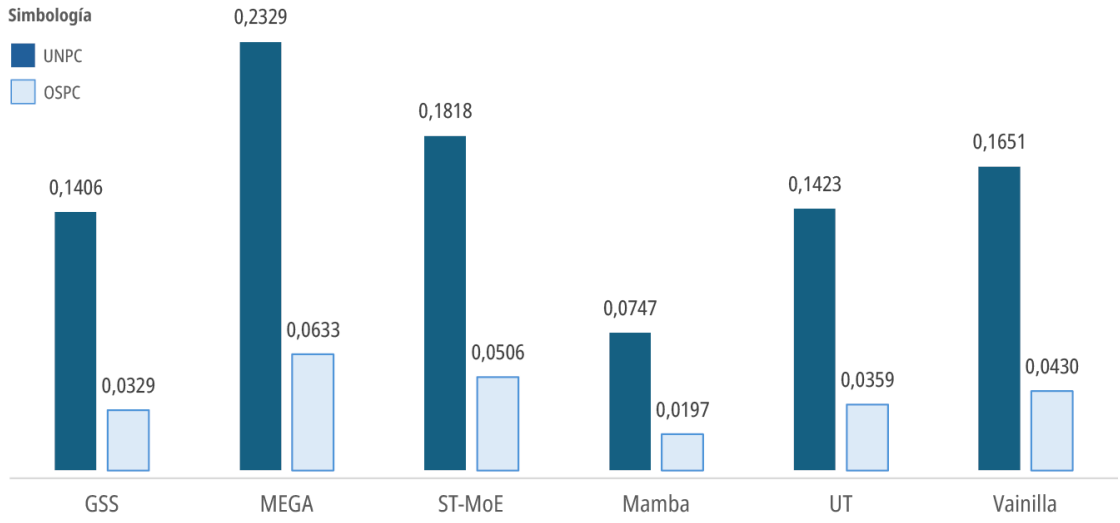
Cuadro No. 6.17. Tiempo promedio de inferencia por registro

Modelo	UNPC	OSPC
GSS	0,1406	0,0329
MEGA	0,2329	0,0633
ST-MoE	0,1818	0,0506
Mamba	0,0747	0,0197
UT	0,1423	0,0359
Vainilla	0,1651	0,0430

Nota: Cifras expresadas en segundos.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.23. Tiempo promedio de inferencia por registro



Nota: Cifras expresadas en segundos.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

La información anterior muestra el tiempo promedio de inferencia por registro, expresado en segundos, tanto en los conjuntos de datos UNPC como OSPC. Esta métrica es esencial para aplicaciones prácticas en las que la velocidad de respuesta es crítica, como sistemas de traducción

automática en tiempo real, asistentes conversacionales o procesamiento en línea de grandes volúmenes de texto.

Además de evaluar la precisión y calidad de las predicciones, es fundamental considerar la **eficiencia temporal** de cada modelo durante el proceso de inferencia, es decir, el tiempo que toma generar una traducción automática una vez entrenado.

En general, se observa que todos los modelos presentan valores de inferencia más altos sobre UNPC que sobre OSPC, lo cual puede explicarse por la longitud promedio mayor de las secuencias en ese corpus y por la complejidad estructural de sus frases. Esta diferencia también puede estar asociada al tamaño del batch de evaluación o al paralelismo efectivo alcanzado en la ejecución.

Los modelos más eficientes en términos de tiempo promedio de inferencia está Mamba, con 0,0747 s (UNPC) y apenas 0,0197 s (OSPC). Del mismo modo ST-MoE, con 0,1818 s (UNPC) y 0,0506 s (OSPC).

Estos modelos presentan una clara ventaja en contextos donde se privilegia la velocidad de respuesta sin comprometer demasiado la precisión. Mamba, en particular, ofrece un excelente equilibrio entre desempeño y eficiencia, destacándose como el modelo con menor tiempo en ambos conjuntos.

En contraste, MEGA es el modelo con el mayor tiempo promedio en UNPC (0,2329 s), seguido por ST-MoE. Este comportamiento puede atribuirse a una mayor complejidad computacional interna, como mecanismos de atención extendidos o capas adicionales de normalización.

Vainilla y UT, por su parte, se mantienen en valores intermedios, con tiempos razonables que podrían considerarse aceptables para aplicaciones que toleren latencias moderadas.

Este análisis demuestra que, si bien modelos como UT, MEGA y ST-MoE ofrecen un rendimiento superior en métricas de precisión e inferencia BLEU, su costo computacional en tiempo de consulta es mayor. Mamba se posiciona como la opción más eficiente para inferencia rápida, con un rendimiento competitivo y un tiempo significativamente menor. Esto lo convierte en una excelente opción para aplicaciones en dispositivos móviles, sistemas embebidos o entornos de producción que requieran alta escalabilidad.

6.6.2.2 Tiempo total de inferencia

Del mismo modo, en una comparativa de soluciones para modelos de traducción automática, el análisis del tiempo total de inferencia del conjunto de prueba es de gran relevancia porque proporciona una visión global de la eficiencia operativa del modelo a escala, complementando al tiempo promedio por registro con una métrica acumulativa que refleja el comportamiento real del sistema ante cargas completas de trabajo.

Este indicador es especialmente útil cuando se evalúa la viabilidad de un modelo en entornos productivos, ya que permite estimar el tiempo necesario para procesar grandes volúmenes de datos, como documentos extensos, bases de datos multilingües, o procesos de traducción masiva. Un modelo que tiene buen desempeño por *query* pero acumula largos tiempos totales puede generar cuellos de botella en sistemas de procesamiento por lotes, traducción de sitios web, o aplicaciones en servidores que atienden múltiples usuarios simultáneamente.

Además, el tiempo total de inferencia permite comparar la escalabilidad real de los modelos, especialmente cuando se combinan múltiples factores como complejidad arquitectónica, uso de recursos y velocidad de procesamiento. Esta métrica resulta también fundamental para dimensionar el coste computacional global, tanto en tiempo como en consumo energético, algo crucial en decisiones de implementación en la nube o sobre hardware dedicado.

Por tanto, el análisis del tiempo total de inferencia del conjunto de prueba no solo permite medir la rapidez de un modelo, sino que también aporta información estratégica para seleccionar soluciones que sean **sostenibles, eficientes y adecuadas para cargas de trabajo reales**, particularmente en sistemas que operan en tiempo continuo o que deben procesar grandes volúmenes de contenido en plazos reducidos.

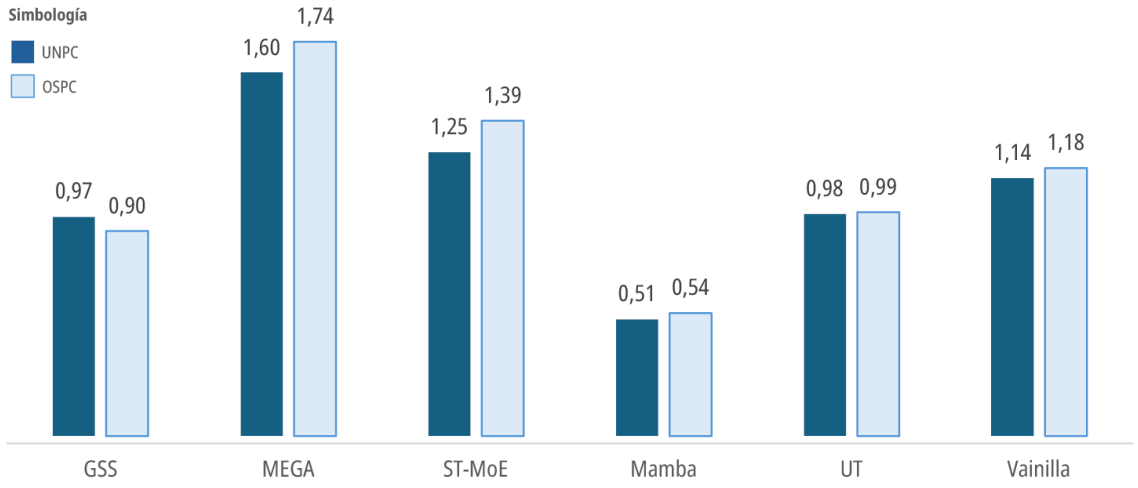
Cuadro No. 6.18. Tiempo total de inferencia del conjunto de datos de prueba

Modelo	UNPC	OSPC
GSS	0,97	0,90
MEGA	1,60	1,74
MoE	1,25	1,39
Mamba	0,51	0,54
UT	0,98	0,99
Vainilla	1,14	1,18

Nota: Cifras expresadas en horas.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

Gráfico No. 6.24. Tiempo total de inferencia



Nota: Cifras expresadas en horas.

Fuente: Comparativa de soluciones, elaboración propia del estudiante.

El cuadro anterior presenta el tiempo total de inferencia requerido por cada arquitectura para procesar el conjunto completo de datos de prueba en los corpus UNPC y OSPC, expresado en horas. Esta métrica es clave para evaluar la viabilidad de los modelos en escenarios de producción a gran escala, donde el rendimiento acumulado puede tener un impacto directo en costos, disponibilidad de infraestructura y tiempos de respuesta al usuario.

El tiempo total requerido para ejecutar todas las consultas de inferencia en un conjunto de datos proporciona una visión integral de la eficiencia operativa de los modelos evaluados. A diferencia del tiempo promedio por consulta, esta métrica tiene mayor relevancia en escenarios donde se requiere el procesamiento de volúmenes grandes de datos o la ejecución por lotes (batch inference), como en sistemas de traducción automática en la nube o motores de preprocesamiento de texto a gran escala.

Entre los modelos más eficientes en términos de tiempo total sobresale Mamba, con apenas 0,51 s (UNPC) y 0,54 s (OSPC). Este resultado reafirma el perfil de Mamba como el modelo más competitivo en términos de eficiencia temporal, superando incluso a modelos más simples como

Vainilla y GSS. Mamba no solo alcanza una alta precisión y robustez en generalización, sino que también demuestra un desempeño óptimo en escenarios operativos.

Por otro lado, los modelos con mayor tiempo total de inferencia sobresalen MEGA, con 1,60 s (UNPC) y 1,74 s (OSPC) y, similarmente, ST-MoE con 1,25 s (UNPC) y 1,39 s (OSPC).

Estas cifras reflejan el costo computacional asociado a su arquitectura más compleja, especialmente cuando se usan mecanismos como mixture-of-experts o capas atencionales pesadas. Si bien estos modelos son sobresalientes en métricas de calidad (BLEU, exactitud), su tiempo total de ejecución puede representar un cuello de botella en entornos con restricciones de recursos o necesidades de escalabilidad.

Los modelos UT y Vainilla presentan tiempos intermedios y consistentes entre corpus. UT, con 0,98 s y 0,99 s, mantiene una buena relación entre precisión y eficiencia, mientras que Vainilla, aunque es más lento (1,14 s y 1,18 s), destaca por su estabilidad y predictibilidad en rendimiento.

Este análisis confirma que Mamba es el modelo más eficiente globalmente, seguido por GSS, aunque este último presenta limitaciones serias en calidad de inferencia. En cambio, MEGA y ST-MoE, si bien son potentes en precisión y BLEU, requieren tiempos significativamente mayores, lo cual puede ser un factor limitante en escenarios de uso continuo o procesamiento paralelo a gran escala.

7 CONCLUSIONES Y TRABAJO FUTURO

7.1 Conclusiones de la comparativa

A partir de la comparativa de las soluciones avanzadas de NMT consideradas en esta comparativa se pueden derivar varias conclusiones importantes para el campo de la traducción automática basada en redes neuronales. Estas conclusiones se centran en la evaluación del rendimiento de los modelos, sus fortalezas, limitaciones y las áreas potenciales de mejora.

La evaluación integral de las arquitecturas de traducción automática, Vainilla (Transformer base), UT, MEGA, ST-MoE, GSS y Mamba, permite concluir que Mamba representa la alternativa más equilibrada y eficiente en términos de rendimiento, costo computacional y adaptabilidad operativa.

Tras la evaluación cuantitativa de seis arquitecturas de traducción automática neuronal (Vainilla, UT, Mamba, ST-MoE, GSS y MEGA) sobre los corpus UNPC y OSPC, es posible extraer una

serie de conclusiones clave respecto al desempeño, eficiencia y capacidad de generalización de cada modelo en diferentes contextos. El análisis abarcó métricas fundamentales como precisión, pérdida, perplejidad, BLEU-4, consumo de recursos y tiempos de inferencia, permitiendo así una valoración integral y comparativa.

7.1.1 Desempeño en precisión y pérdida

Los modelos UT, ST-MoE y MEGA destacan sistemáticamente como los más precisos, alcanzando los valores más altos tanto en entrenamiento como en validación. Sin embargo, este alto desempeño viene acompañado de brechas negativas considerables entre ambas fases, evidenciando una tendencia al sobreajuste, especialmente sobre el corpus OSPC.

Por el contrario, Mamba y Vainilla lograron un balance más estable entre entrenamiento y validación, con pequeñas diferencias o incluso mejoras en validación, lo que sugiere una mayor capacidad de generalización. GSS, aunque mostró brechas positivas, sufre de una precisión absoluta baja, limitando su utilidad práctica.

7.1.2 Perplejidad y estabilidad de aprendizaje

El análisis de la perplejidad confirmó que UT, Mamba y ST-MoE son los modelos con menor incertidumbre al momento de predecir secuencias, lo que refleja su solidez estructural. En contraste, GSS presentó valores extremadamente altos e inestables de perplejidad, lo que evidencia problemas graves en su convergencia y capacidad de aprendizaje sobre secuencias complejas.

7.1.3 Evaluación en inferencia (BLEU-4)

Según la métrica BLEU-4, que mide la similitud de las traducciones automáticas respecto a referencias humanas, ST-MoE y UT lideran el ranking en ambos corpus, aunque todos los modelos muestran una caída esperada al pasar de UNPC a OSPC. Sorprendentemente, Vainilla y Mamba obtienen resultados competitivos en OSPC, mostrando que arquitecturas más ligeras pueden ofrecer un rendimiento aceptable incluso en contextos complejos.

7.1.4 Tiempos de inferencia y eficiencia computacional

En términos de eficiencia operativa, Mamba se posiciona como el modelo más eficiente, tanto en tiempo promedio por consulta como en tiempo total de inferencia. Esto lo convierte en una opción idónea para aplicaciones en tiempo real o dispositivos con recursos limitados. GSS, aunque también

rápido, presenta limitaciones de calidad. Por el contrario, MEGA y ST-MoE, a pesar de su alto rendimiento, requieren más tiempo total de procesamiento, lo que puede restringir su aplicabilidad en escenarios de alto volumen.

7.1.5 Memoria y parámetros entrenables

Aunque el consumo de memoria no mostró diferencias marcadas entre modelos, sí se observaron diferencias importantes en la cantidad de parámetros. UT y ST-MoE fueron los modelos más complejos, lo cual se traduce en mayor precisión pero también en mayor costo computacional, tanto en entrenamiento como en despliegue.

7.1.6 Recomendaciones finales

Esta evaluación ofrece así un panorama completo y comparativo que permite seleccionar el modelo más adecuado según el contexto de aplicación, ya sea en términos de precisión, eficiencia, estabilidad o escalabilidad.

Si se prioriza precisión absoluta y calidad de traducción, modelos como ST-MoE y UT son recomendables, siempre que se cuente con suficiente capacidad computacional y mecanismos para mitigar el sobreajuste.

Por otro lado, si el objetivo es un balance entre rendimiento, eficiencia y generalización, Mamba se posiciona como la opción más robusta y versátil.

Para aplicaciones educativas o de prototipado, Vainilla representa una alternativa predecible, estable y relativamente eficiente.

Finalmente, GSS no es recomendable en su configuración actual, dada su inestabilidad, baja precisión y comportamiento errático en múltiples métricas.

7.2 Conclusiones de la revisión del estado del arte

Con base en la teoría presentada y los resultados empíricos obtenidos en esta comparativa de arquitecturas para traducción automática, se pueden establecer las siguientes conclusiones fundamentadas tanto en evidencia experimental como en el marco teórico:

Los Transformers continúan siendo el estándar sólido, pero las variantes superan en eficiencia y precisión en contextos específicos. Los resultados empíricos coinciden con lo señalado por

Vaswani et al. (2017) y (So et al., 2019): los Transformers siguen siendo una base robusta y confiable, como lo demuestra el modelo Vainilla en esta comparativa. Sin embargo, arquitecturas como UT y MEGA, alineadas con el concepto de Transformers evolucionados, mostraron mejoras considerables en precisión y métricas BLEU, aunque con un coste computacional más alto. Esto valida la noción de que las iteraciones estructurales pueden optimizar el rendimiento a costa de recursos.

El enfoque Mixture of Experts (MoE) demuestra gran escalabilidad y buena calidad de traducción. Tal como lo afirma Zoph et al. (2022), MoE es especialmente eficaz al distribuir el cómputo entre expertos especializados, y esto se refleja en su alto rendimiento en métricas BLEU y precisión. Sin embargo, el tiempo total de entrenamiento y la inferencia no fueron tan competitivos como en otros modelos, lo que sugiere que su eficiencia estructural se aprovecha mejor en contextos de entrenamiento masivo y no necesariamente en tareas con restricciones operativas severas.

Los mecanismos de gating y atención presentan ventajas en tareas con secuencias complejas, pero su efectividad práctica es variable. Aunque la teoría sugiere que arquitecturas como Gated State Spaces (GSS) y Moving Average Gated Attention son idóneas para manejar dependencias largas (Mehta et al., 2022; Ma et al., 2022), los resultados experimentales muestran un desempeño considerablemente bajo en GSS, tanto en pérdida como en precisión y BLEU. Esto sugiere que la implementación práctica de estos enfoques requiere una mayor madurez o combinación con otras técnicas para alcanzar su potencial.

El modelo Mamba, basado en un Space-State Model simplificado, confirma que es posible reducir complejidad sin comprometer el rendimiento. Tal como argumenta Smith et al. (2022), las representaciones compactas permiten una generalización eficaz con bajo coste computacional. Mamba fue el modelo que mejor balance logró entre precisión, calidad de **traducción, eficiencia de entrenamiento e inferencia**, lo que lo convierte en una excelente materialización de los beneficios teóricos propuestos por este tipo de arquitectura simplificada.

Las métricas de calidad de traducción, como BLEU, reflejan diferencias importantes que se alinean con los enfoques estructurales. Modelos como MoE, MEGA y UT dominaron en BLEU, especialmente en el corpus UNPC, lo que valida que las variantes del Transformer pueden mejorar la calidad de traducción en corpus estructurados. No obstante, Mamba igualó o superó estos modelos en OSPC, lo que demuestra su capacidad de adaptación a contextos más ruidosos sin comprometer eficiencia.

El tiempo de entrenamiento e inferencia es un diferenciador clave para la implementación práctica. Las diferencias observadas respaldan lo planteado en la teoría: arquitecturas como MoE y Mamba muestran eficiencia operativa, pero solo Mamba combina esta ventaja con tiempos de inferencia extremadamente bajos, posicionándose como la mejor opción en tareas con exigencias de respuesta rápida o procesamiento a gran escala.

Las LSTM extendidas no fueron evaluadas en esta comparativa, pero los resultados sugieren que su integración con arquitecturas modernas podría ser valiosa. Aunque las Extended LSTM propuestas por Beck et al. (2024) no se incluyeron en los experimentos, la evidencia sugiere que combinar su capacidad de modelar continuidad temporal con arquitecturas eficientes como Mamba o MoE podría generar soluciones híbridas con mayor capacidad adaptativa.

Los enfoques híbridos ofrecen una vía prometedora para avanzar en eficiencia y rendimiento. A la luz de los datos y la teoría, resulta evidente que la combinación de mecanismos de atención, gating y expertos especializados, como los que presentan MoE o Mamba, representan una dirección valiosa para futuras investigaciones. En particular, modelos como Mamba podrían beneficiarse de la incorporación controlada de componentes expertos o atencionales sin perder su eficiencia base.

A pesar de que numerosas arquitecturas han sido propuestas como mejoras o extensiones al modelo Transformer original (Vaswani et al., 2017), dicho modelo no ha sido desplazado como estándar de facto en tareas de traducción automática y procesamiento del lenguaje natural (PLN). Esta permanencia se debe a una combinación de factores que van más allá de métricas individuales de rendimiento:

El Transformer original ha demostrado una estabilidad excepcional en múltiples tareas y corpus, lo que le ha permitido consolidarse como una arquitectura confiable para producción. Su comportamiento predecible, incluso ante secuencias largas o con estructuras gramaticales diversas, lo hace atractivo frente a arquitecturas más novedosas, pero menos probadas.

Existe un ecosistema robusto de librerías, frameworks, modelos preentrenados y documentación optimizada para Transformers clásicos. Arquitecturas como BERT, GPT, T5, MarianMT, entre otros, se basan en adaptaciones del Transformer base, lo cual facilita la reutilización de recursos. Cambiar a arquitecturas completamente distintas implicaría rehacer pipelines de entrenamiento, modificar optimizadores y, en muchos casos, sacrificar compatibilidad con infraestructuras existentes.

Si bien modelos como UT, Mamba, ST-MoE o MEGA presentan innovaciones importantes (recurrentes, mixturas de expertos, cambios en la atención), estas no siempre generan mejoras sustanciales en contextos reales, o lo hacen a costa de incrementar significativamente la complejidad computacional, el número de parámetros o el tiempo de inferencia. En ambientes con restricciones operativas, estos factores pueden ser decisivos.

Las arquitecturas sucesoras del Transformer se encuentran aún en una fase de exploración y validación académica. No existe un consenso unánime sobre cuál de ellas representa una mejora generalizada. Algunas sobresalen en tareas específicas o corpus particulares (por ejemplo, UT en secuencias largas), pero no presentan ventajas sistemáticas en todos los escenarios, lo cual limita su adopción masiva.

El modelo Transformer puede ser ajustado, regularizado y adaptado para contextos específicos con técnicas como pruning, distillation, entrenamiento multitarea y embeddings posicionales aprendibles, lo cual le permite competir de forma muy cercana con propuestas más recientes, sin requerir arquitecturas radicalmente diferentes.

La razón por la cual las arquitecturas sucesoras aún no han sustituido al Transformer original no radica en una falta de mérito técnico, sino en un balance desfavorable entre innovación, estabilidad, eficiencia y adopción práctica. En contextos de investigación, estas propuestas son valiosas y han empujado los límites del campo; sin embargo, en aplicaciones productivas, el Transformer sigue ofreciendo una relación costo-beneficio difícil de superar.

7.3 Trabajo futuro

Como se mencionó el objetivo de este trabajo no es presentar un nuevo algoritmo de entrenamiento, más bien busca indagar sobre diferentes aspectos que pueden conducir a una forma más económicamente viable de lograr el entrenamiento de modelos de redes profundas.

Un trabajo posterior podría enfocarse en abordar estas técnicas y evaluar su aplicabilidad y beneficios en el entrenamiento y la implementación de modelos de inteligencia artificial en un entorno de hardware de bajo costo, mixto o completamente especializado.

En el contexto de una tesis doctoral, podría resultar de interés desarrollar los diferentes enfoques estudiados acá y proponer un modelo unificado que combine las diferentes técnicas estudias y proponga una arquitectura que optimice los procesos de entrenamiento de redes

neuronales artificiales enfatizando el bajo costo y la reutilización de componentes de hardware de consumo.

Estas recomendaciones ofrecen direcciones para investigar en futuros trabajos sobre NMT, explorando combinaciones innovadoras entre arquitecturas avanzadas como Transformers, MoE, GSS, y LSTM extendidas. Los enfoques híbridos y las técnicas emergentes como el uso de gating, modelos simplificados y compresión de modelos podrían llevar a mejoras significativas en la traducción automática, tanto en términos de eficiencia como de rendimiento. La investigación futura podría centrarse en probar estas combinaciones en diferentes conjuntos de datos y configuraciones de hardware para evaluar su efectividad y escalabilidad.

A partir de la comparativa de las arquitecturas avanzadas de Neural Machine Translation (NMT) mencionadas es posible identificar varias direcciones prometedoras para futuros trabajos de investigación. Las recomendaciones que se presentan a continuación no solo consideran el rendimiento de las arquitecturas individuales, sino también las oportunidades para combinar enfoques que podrían ofrecer mejoras en términos de eficiencia, escalabilidad, y capacidad de generalización.

7.3.1 Combinación de Mixture of Experts (MoE) con Gated State Spaces (GSS)

Una dirección interesante sería combinar ST-MoE (Mixture of Experts) con la arquitectura de Gated State Spaces (GSS). ST-MoE introduce módulos especializados que se activan en función de la tarea específica, lo que permite un uso más eficiente de los recursos. Por otro lado, GSS ofrece una forma avanzada de controlar el flujo de información a través de mecanismos de gating que pueden mejorar la captura de relaciones complejas en secuencias largas.

MoE se destaca en escenarios donde se busca reducir el costo computacional al activar solo una parte de la red, mientras que GSS mejora el control y filtrado de información relevante. La combinación de estos dos enfoques podría mejorar tanto la eficiencia como la calidad de las traducciones, especialmente en secuencias largas o complejas. La combinación de MoE con técnicas de gating ha sido explorada en otros trabajos, como el modelo de Switch Transformer (Fedus et al., 2021), que integra ideas de gating para activar expertos específicos.

7.3.2 Incorporación de Atención Promediada en Modelos Transformer

La introducción de mecanismos de atención promediada, como el *Moving Average Equipped Gated Attention* (Ma et al., 2022), podría mejorar las arquitecturas *Transformer* al suavizar la información a través de ventanas temporales y mejorar la estabilidad en el entrenamiento de secuencias largas. Este enfoque puede ser beneficioso cuando se integra en *Evolved Transformers*, que ya han optimizado varias partes del modelo base *Transformer*.

La atención promediada permite mejorar la robustez del modelo al reducir el ruido y mitigar las fluctuaciones rápidas en el proceso de entrenamiento. Esto podría hacer que los Transformers evolucionados sean aún más efectivos para tareas de traducción de textos largos o con dependencia a largo plazo.

La combinación de mecanismos de atención promediada con Transformers ha sido explorada en algunos trabajos, como Reformer (Kitaev et al., 2020), que utiliza hashing local para mejorar la eficiencia del mecanismo de atención.

7.3.3 Integración de Extended LSTM con Arquitecturas Transformer

Aunque los Transformers han superado a las LSTM en muchos casos, el uso de Extended LSTM (Beck et al., 2024) podría proporcionar beneficios si se integran en capas híbridas junto con arquitecturas de Transformers. Las LSTM extendidas ofrecen un control adicional sobre la memoria a largo plazo y podrían complementar las limitaciones de los Transformers en la gestión de dependencias a largo plazo.

Mientras los Transformers destacan por su capacidad de procesamiento paralelo y su mecanismo de atención, las LSTM pueden ser más efectivas en la captura de relaciones temporales continuas a lo largo de la secuencia. El enfoque híbrido podría ofrecer lo mejor de ambos mundos: la potencia computacional del Transformer junto con la capacidad de las LSTM para recordar información pasada más allá de las ventanas de atención.

Trabajos como el de Combinations of RNN and Transformer (Liu et al., 2020) han explorado este enfoque híbrido, combinando Transformers con redes recurrentes para mejorar la traducción en dominios de lenguaje natural.

7.3.4 Exploración de Modelos Auto-regresivos Híbridos

Otra opción de investigación sería investigar la combinación de Universal Transformers con modelos *Simplified Space-State*. Los Universal Transformers presentan una forma de iterar sobre la secuencia de entrada de manera recurrente, aplicando el mismo mecanismo de atención en cada paso, lo que los convierte en una versión más general de los Transformers.

La capacidad de los *Simplified Space-State Models* para simplificar la estructura temporal y mejorar la capacidad de modelar secuencias largas podría hacer sinergia con la naturaleza iterativa de los Universal Transformers, potenciando el aprendizaje de patrones complejos en la traducción automática.

Universal Transformers se han combinado con técnicas recurrentes avanzadas en otras investigaciones, como los modelos *Recurrent Memory Networks* (Rae et al., 2021).

7.3.5 Hibridación de Modelos con MoE y Transformers Evolucionados

La combinación de la arquitectura *Evolved Transformer* con el enfoque MoE (ST-MoE) podría ser un camino interesante para optimizar tanto la eficiencia del modelo como su capacidad de generalización. El *Evolved Transformer* ya introduce mejoras basadas en búsquedas neuronales automáticas, mientras que MoE permitiría distribuir los recursos de manera más eficiente al activar expertos solo cuando sea necesario.

El *Evolved Transformer* optimiza la arquitectura *Transformer* tradicional, mientras que MoE mejora la escalabilidad y eficiencia. Combinando ambos enfoques, se podría lograr un modelo que no solo es más eficiente en términos computacionales, sino que también es capaz de manejar tareas de traducción complejas con menos recursos.

Modelos como el *Switch Transformer* (Fedus et al., 2021) ya combinan ideas de MoE con Transformers para reducir el costo computacional sin sacrificar la precisión.

7.3.6 Implementación de Técnicas de Compresión de Modelos

Un área emergente en la investigación de NMT es la compresión de modelos mediante *distillation* o *pruning*, lo que podría aplicarse a modelos como ST-MoE y *Gated State Spaces*. Dado que los modelos de gran tamaño pueden ser prohibitivamente costosos, reducir su tamaño sin perder precisión es un enfoque relevante.

Comprimir modelos que ya utilizan *gating* o expertos como MoE podría mejorar la eficiencia sin comprometer la precisión. Esta combinación sería especialmente útil en contextos donde los recursos computacionales son limitados, como en dispositivos móviles.

Trabajos recientes como DistilBERT (Sanh et al., 2019) han mostrado cómo se puede comprimir modelos preentrenados sin perder precisión significativa, lo que podría aplicarse en modelos de NMT de gran escala.

8 REFERENCIAS BIBLIOGRAFICAS

Anastasopoulos, A., & Chiang, D. (2018). Tied Multitask Learning for Neural Speech Translation. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 82–91. https://doi.org/10.18653/v1/N18-1008

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., & Hochreiter, S. (2024). *xLSTM: Extended Long Short-Term Memory*.

Bhojanapalli, S., Yun, C., Rawat, A. S., Reddi, S., & Kumar, S. (2020). Low-rank bottleneck in multi-head attention models. *37th International Conference on Machine Learning, ICML 2020*, *PartF168147-2*.

Bishop, C. M., & Bishop, H. (2024). *Deep Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-45468-4

Brauwers, G., & Frasincar, F. (2023). A General Survey on Attention Mechanisms in Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*(4). https://doi.org/10.1109/TKDE.2021.3126456

Chaudhari, S., Mithal, V., Polatkan, G., & Ramanath, R. (2021). An Attentive Survey of Attention Models. *ACM Transactions on Intelligent Systems and Technology*, *12*(5). https://doi.org/10.1145/3465055

Csordás, R., Irie, K., Schmidhuber, J., Potts, C., & Manning, C. D. (2024). *MoEUT: Mixture-of-Experts Universal Transformers*. https://arxiv.org/abs/2405.16039

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, Ł. (2019). *Universal Transformers*. https://arxiv.org/abs/1807.03819

Dzmitry Bahdanau, Kyung Hyun Cho, & Yoshua Bengio. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Eisenstein, J. (2019). *Introduction to Natural Language Processing*. MIT Press. https://books.google.co.cr/books?id=72yuDwAAQBAJ

Fedus, W., Zoph, B., & Shazeer, N. (2022). *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. https://arxiv.org/abs/2101.03961

Firth, J. R. (1957). A Synopsis of Linguistic Theory 1930-55. In *Studies in Linguistic Analysis: Special Volume of the Philological Society*.

Gammerman, A., Vovk, V., & Vapnik, V. (2013). *Learning by Transduction*.

Graves, A. (2012, November 14). *Sequence Transduction with Recurrent Neural Networks*.

Grefenstette, E., Hermann, K. M., Suleyman, M., & Blunsom, P. (2015). *Learning to Transduce with Unbounded Memory*.

Gu, A., & Dao, T. (2024). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. https://arxiv.org/abs/2312.00752

Gu, A., Dao, T., Ermon, S., Rudra, A., & Re, C. (2020). *HiPPO: Recurrent Memory with Optimal Polynomial Projections*.

Gu, A., Goel, K., & Ré, C. (2021). *Efficiently Modeling Long Sequences with Structured State Spaces*.

Harris, Z. S. (1954). Distributional Structure. WORD, *10*(2–3). https://doi.org/10.1080/00437956.1954.11659520

Jaitly, N., Sussillo, D., Le, Q. V., Vinyals, O., Sutskever, I., & Bengio, S. (2015). *A Neural Transducer*.

Kamath, U., Graham, K. L., & Emara, W. (2022). *Transformers for Machine Learning*. Chapman and Hall/CRC. https://doi.org/10.1201/9781003170082

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*.

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). *Reformer: The Efficient Transformer*. https://arxiv.org/abs/2001.04451

Koehn, P. (2020). *Neural Machine Translation*. Cambridge University Press. https://doi.org/10.1017/9781108608480

Le, H., Pino, J., Wang, C., Gu, J., Schwab, D., & Besacier, L. (2020). Dual-decoder Transformer for Joint Automatic Speech Recognition and Multilingual Speech Translation. *COLING 2020 - 28th International Conference on Computational Linguistics, Proceedings of the Conference*. https://doi.org/10.18653/v1/2020.coling-main.314

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., Abend, O., Alon, R., Asida, T., Bergman, A., Glozman, R., Gokhman, M., Manevich, A., Ratner, N., Rozen, N., … Shoham, Y. (2024). *Jamba: A Hybrid Transformer-Mamba Language Model*. https://arxiv.org/abs/2403.19887

Lin, P., Martins, A. F. T., & Schütze, H. (2024). *A Recipe of Parallel Corpora Exploitation for Multilingual Large Language Models*.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, *3*. https://doi.org/10.1016/j.aiopen.2022.10.001

Lison, P., & Tiedemann, J. (2016). OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 923–929). European Language Resources Association (ELRA). https://aclanthology.org/L16-1147

Liu, Z., Lin, Y., & Su, M. (2023). Representation Learning for Natural Language Processing. In *Representation Learning for Natural Language Processing*. https://doi.org/10.1007/978-981-15-5573-2

Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J., & Zettlemoyer, L. (2022). *Mega: Moving Average Equipped Gated Attention*.

Mehta, H., Gupta, A., Cutkosky, A., & Neyshabur, B. (2022). *Long Range Language Modeling via Gated State Spaces*.

Pióro, M., Ciebiera, K., Król, K., Ludziejewski, J., Krutul, M., Krajewski, J., Antoniak, S., Miłoś, P., Cygan, M., & Jaszczur, S. (2024). *MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts*. https://arxiv.org/abs/2401.04081

Prince, S. J. D. (2023). *Understanding Deep Learning*. MIT Press. http://udlbook.com

Smith, J. T. H., Warrington, A., & Linderman, S. W. (2022). *Simplified State Space Layers for Sequence Modeling*.

So, D. R., Liang, C., & Le, Q. V. (2019). *The Evolved Transformer*.

Sperber, M., Setiawan, H., Gollan, C., Nallasamy, U., & Paulik, M. (2020). Consistent Transcription and Translation of Speech. *Transactions of the Association for Computational Linguistics*, *8*, 695–709. https://doi.org/10.1162/tacl_a_00340

Sun, J., Liu, X., Mei, X., Kılıç, V., Plumbley, M. D., & Wang, W. (2023). Dual Transformer Decoder based Features Fusion Network for Automated Audio Captioning. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, *2023-August*. https://doi.org/10.21437/Interspeech.2023-943

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *4*(January).

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2022). Efficient Transformers: A Survey. *ACM Computing Surveys*, *55*(6). https://doi.org/10.1145/3530811

Vardasbi, A., Pires, T. P., Schmidt, R. M., & Peitz, S. (2023a). *State Spaces Aren't Enough: Machine Translation Needs Attention*.

Vardasbi, A., Pires, T. P., Schmidt, R., & Peitz, S. (2023b). State Spaces Aren't Enough: Machine Translation Needs Attention. In M. Nurminen, J. Brenner, M. Koponen, S. Latomaa, M. Mikhailov, F. Schierl, T. Ranasinghe, E. Vanmassenhove, S. A. Vidal, N. Aranberri, M. Nunziatini, C. P. Escart\'\in, M. Forcada, M. Popovic, C. Scarton, & H. Moniz (Eds.), *Proceedings of the 24th Annual Conference of the European Association for Machine Translation* (pp. 205–216). European Association for Machine Translation. https://aclanthology.org/2023.eamt-1.20

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-December*.

Wang, S., Tu, Z., Tan, Z., Wang, W., Sun, M., & Liu, Y. (2021). *Language Models are Good Translators*.

Wang, Z. (2023). *Learning Transductions and Alignments with RNN Seq2seq Models*.

Xiao, T., & Zhu, J. (2023). *Introduction to Transformers: an NLP Perspective*.

Yang, S., Wang, B., Shen, Y., Panda, R., & Kim, Y. (2024). *Gated Linear Attention Transformers with Hardware-Efficient Training*. https://arxiv.org/abs/2312.06635

Yu, Z., Wang, J., Yu, L.-C., & Zhang, X. (2022). Dual-Encoder Transformers with Cross-modal Alignment for Multimodal Aspect-based Sentiment Analysis. *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, *1*.

Zhang, Z., Shao, N., Gao, C., Miao, R., Yang, Q., & Shao, J. (2022). Mixhead: Breaking the low-rank bottleneck in multi-head attention language models. *Knowledge-Based Systems*, *240*, 108075. https://doi.org/10.1016/j.knosys.2021.108075

Zhou, Z. H. (2012). Ensemble methods: Foundations and algorithms. In *Ensemble Methods: Foundations and Algorithms*. https://doi.org/10.1201/b12207

Ziemski, M., Junczys-Dowmunt, M., & Pouliquen, B. (2016). The United Nations Parallel Corpus v1.0. In N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 3530–3534). European Language Resources Association (ELRA). https://aclanthology.org/L16-1561

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., & Fedus, W. (2022). *ST-MoE: Designing Stable and Transferable Sparse Expert Models*.

9 ANEXOS

Anexo No. 1. Arquitecturas de modelos y sus hiperparámetros

Modelo 1: Vanilla Transformer (Clase: NMT_Vashwani)

- **Descripción:** Basado en la arquitectura Transformer original de Vaswani et al. (2017).
- **Parámetros Principales (instanciados):**
    - num_encoder_layers: 6
    - num_decoder_layers: 6
    - emb_size (d_model): 512
    - nhead (número de cabezas de atención): 8
    - ff_dim (dimensión de la capa feedforward): 512
    - dropout: 0.1 (valor por defecto en la definición de la clase nn.Transformer interna)
    - lr: 0.0001 (Variable global TFMR_LR)
- **Instancias:** model_Vainilla_UNPC y model_Vainilla_OSPC (difieren en src_vocab_size, tgt_vocab_size).

Modelo 2: Universal Transformer (UT) (Clase: UT_NMT)

- **Descripción:** Implementación de un Transformer Recurrente (Universal Transformer).
- **Parámetros Principales (definidos en XT_Config dentro de la clase):**
    - emb_dim: 512
    - hidden_dim: 1024
    - pff_dim: 1024
    - n_layers: 4
    - n_heads: 8
    - dropout_ratio: 0.1
    - max_len: Variable (UN_MAX_LEN o OS_MAX_LEN según dataset)

    - pad_id: 0
    - lr: 0.0001 (Variable global TFMR_LR)
- **Instancias:** model_UT_UNPC y model_UT_OSPC (difieren en vocab_size y max_length).

Modelo 3: S5 (Simplified Space-State Model - SSM) (Clase: SSM_NMT)

- **Descripción:** Modelo basado en espacios de estado estructurados (SSM), utilizando la capa Mamba.
- **Parámetros Principales (instanciados en SSM_NMT):**
    - d_model: 512
    - n: 256 (Parece no usarse directamente en las capas internas según el código, pero está en el init)
    - block_count: 4 (Usado como expand en la capa Mamba interna)
    - dropout: 0.3
    - max_length: Variable (UN_MAX_LEN o OS_MAX_LEN según dataset)
    - lr: 0.0001 (Variable global TFMR_LR)
- **Parámetros Internos (Capa Mamba en SSM_Encoder/Decoder):**
    - d_state: 16 (Factor de expansión del estado SSM)
    - d_conv: 4 (Ancho de la convolución local)
    - expand: 2 (Factor de expansión del bloque)
- **Parámetros Internos (Capa MultiHeadAttention):**
    - n_heads: 8
- **Instancias:** model_SSM_UNPC y model_SSM_OSPC (difieren en vocab_size y max_length).

Modelo 4: ST-MoE Transformer (Clase: MoE_NMT)

- **Descripción:** Transformer que incorpora un bloque Mixture-of-Experts (MoE) disperso.
- **Parámetros Principales (instanciados o definidos internamente):**
    - d_model: 512

    - n_heads (para MultiHeadAttention): 8
    - num_experts (para MoE): 8
    - gating_top_n (para MoE): 2
    - dropout: 0.1 (definido en las capas internas)
    - max_seq_length: Variable (UN_MAX_LEN o OS_MAX_LEN según dataset)
    - lr: 0.0001 (Variable global TFMR_LR)
- **Instancias:** model_MOE_UNPC y model_MOE_OSPC (difieren en vocab_size y max_length).

Modelo 5: Gated State Spaces (GSS) (Clase: GSS_NMT)

- **Descripción:** Modelo basado en Gated State Spaces.
- **Parámetros Principales (instanciados o definidos internamente):**
    - d_model: 512
    - n_heads (para MultiHeadAttention): 8
    - depth (número de capas GSS): 3
    - dss_kernel_H (para GSS): 512
    - dss_kernel_N (para GSS): 256
    - dim_expansion_factor (para GSS): 4
    - dropout: 0.1 (definido en las capas internas)
    - max_seq_length: Variable (UN_MAX_LEN o OS_MAX_LEN según dataset)
    - lr: 0.0001 (Variable global TFMR_LR)
- **Instancias:** model_GSS_UNPC y model_GSS_OSPC (difieren en vocab_size y max_length).

Modelo 6: Moving Average Equipped Gated Attention (MEGA) (Clase: MEGA_NMT)

- **Descripción:** Modelo que utiliza capas MEGA (Moving Average Equipped Gated Attention).
- **Parámetros Principales (instanciados o definidos internamente):**
    - d_model: 512

    - n_heads (para MultiHeadAttention): 8
    - depth (número de capas MegaLayer): 3
    - ema_heads (para MegaLayer): 16
    - attn_dim_qk (para MegaLayer): 64
    - attn_dim_value (para MegaLayer): 256
    - dropout: 0.1 (definido en las capas internas)
    - max_seq_length: Variable (UN_MAX_LEN o OS_MAX_LEN según dataset)
    - lr: 0.0001 (Variable global TFMR_LR)
- **Instancias:** model_MEGA_UNPC y model_MEGA_OSPC (difieren en vocab_size y max_length).

Anexo No. 2. Resultados de pérdida, precisión y perplejidad

model_name	epoch	dataset	trn_loss	trn_accm	trn_pplx	val_loss	val_accm	val_pplx
Vainilla	1	UNPC	5,0538	0,2222	156,6165	4,4282	0,2929	83,7805
Vainilla	2	UNPC	4,2816	0,3075	72,3561	4,0245	0,3397	55,9523
Vainilla	3	UNPC	3,9974	0,3444	54,4564	3,8199	0,3710	45,5996
Vainilla	4	UNPC	3,8323	0,3674	46,1686	3,6826	0,3906	39,7496
Vainilla	5	UNPC	3,7302	0,3845	41,6874	3,6835	0,3999	39,7854
Vainilla	6	UNPC	3,7692	0,3968	43,3454	3,7371	0,4094	41,9761
Vainilla	7	UNPC	3,7551	0,4080	42,7385	3,6738	0,4242	39,4013
Vainilla	8	UNPC	3,7162	0,4189	41,1079	3,6507	0,4332	38,5016
Vainilla	9	UNPC	3,6816	0,4288	39,7099	3,6582	0,4362	38,7915
Vainilla	10	UNPC	3,6435	0,4379	38,2254	3,6265	0,4436	37,5811
Vainilla	11	UNPC	3,6022	0,4467	36,6788	3,5427	0,4586	34,5601
Vainilla	12	UNPC	3,5656	0,4545	35,3607	3,5214	0,4642	33,8318
Vainilla	13	UNPC	3,5355	0,4601	34,3122	3,5089	0,4697	33,4115
Vainilla	14	UNPC	3,5067	0,4665	33,3381	3,4837	0,4725	32,5800
Vainilla	15	UNPC	3,4778	0,4724	32,3884	3,4539	0,4810	31,6235
Vainilla	16	UNPC	3,446	0,4778	31,3746	3,4276	0,4848	30,8026
Vainilla	17	UNPC	3,4163	0,4829	30,4565	3,4103	0,4868	30,2743
Vainilla	18	UNPC	3,3905	0,4871	29,6808	3,3875	0,4920	29,5919
Vainilla	19	UNPC	3,3642	0,4911	28,9104	3,3689	0,4942	29,0466
Vainilla	20	UNPC	3,3369	0,4943	28,1318	3,3428	0,4971	28,2983
Vainilla	1	OSPC	5,0053	0,2919	149,2018	4,3120	0,3627	74,5895
Vainilla	2	OSPC	4,1717	0,3791	64,8256	3,8852	0,4150	48,6767
Vainilla	3	OSPC	3,8821	0,4141	48,5260	3,6807	0,4418	39,6742
Vainilla	4	OSPC	3,7055	0,4354	40,6704	3,5462	0,4593	34,6813
Vainilla	5	OSPC	3,5785	0,4506	35,8198	3,4502	0,4717	31,5067
Vainilla	6	OSPC	3,4733	0,4633	32,2430	3,3682	0,4813	29,0262
Vainilla	7	OSPC	3,3896	0,4741	29,6541	3,3143	0,4892	27,5031
Vainilla	8	OSPC	3,3504	0,4829	28,5141	3,3541	0,4942	28,6198
Vainilla	9	OSPC	3,3550	0,4907	28,6456	3,3443	0,5011	28,3407
Vainilla	10	OSPC	3,3312	0,4982	27,9719	3,3437	0,5062	28,3237
Vainilla	11	OSPC	3,3421	0,4994	28,2784	3,3366	0,5094	28,1233
Vainilla	12	OSPC	3,3258	0,5049	27,8212	3,3334	0,5136	28,0335
Vainilla	13	OSPC	3,3109	0,5102	27,4098	3,3248	0,5169	27,7934
Vainilla	14	OSPC	3,2983	0,5149	27,0666	3,3318	0,5204	27,9887
Vainilla	15	OSPC	3,2700	0,5220	26,3113	3,3339	0,5222	28,0475
Vainilla	16	OSPC	3,2552	0,5264	25,9248	3,3251	0,5245	27,8018
Vainilla	17	OSPC	3,2388	0,5307	25,5031	3,3335	0,5262	28,0363
Vainilla	18	OSPC	3,2226	0,5347	25,0933	3,3254	0,5290	27,8101
Vainilla	19	OSPC	3,0968	0,5569	22,1270	3,2924	0,5403	26,9074
Vainilla	20	OSPC	3,0838	0,5623	21,8412	3,2961	0,5410	27,0071
UT	1	UNPC	4,0222	0,3726	55,8238	3,2661	0,4754	26,2089

model_name	epoch	dataset	trn_loss	trn_accm	trn_pplx	val_loss	val_accm	val_pplx
UT	2	UNPC	3,0724	0,5014	21,5937	2,8494	0,5382	17,2774
UT	3	UNPC	2,7442	0,5531	15,5522	2,6285	0,5758	13,8530
UT	4	UNPC	2,5576	0,5854	12,9048	2,5344	0,5979	12,6089
UT	5	UNPC	2,5206	0,6058	12,4361	2,5352	0,6131	12,6190
UT	6	UNPC	2,4694	0,6237	11,8154	2,5099	0,6265	12,3037
UT	7	UNPC	2,4233	0,6379	11,2830	2,4798	0,6367	11,9389
UT	8	UNPC	2,3821	0,6495	10,8276	2,4622	0,6444	11,7306
UT	9	UNPC	2,346	0,6590	10,4437	2,4481	0,6500	11,5663
UT	10	UNPC	2,3135	0,6667	10,1097	2,4273	0,6558	11,3283
UT	11	UNPC	2,2787	0,6740	9,7640	2,4107	0,6589	11,1418
UT	12	UNPC	2,2504	0,6797	9,4915	2,3932	0,6633	10,9485
UT	13	UNPC	2,2204	0,6852	9,2110	2,3850	0,6651	10,8591
UT	14	UNPC	2,195	0,6895	8,9800	2,3679	0,6680	10,6750
UT	15	UNPC	2,1671	0,6942	8,7329	2,3617	0,6689	10,6090
UT	16	UNPC	2,1436	0,6984	8,5301	2,3388	0,6721	10,3688
UT	17	UNPC	2,1185	0,7021	8,3187	2,3438	0,6720	10,4208
UT	18	UNPC	2,0955	0,7056	8,1295	2,3309	0,6745	10,2872
UT	19	UNPC	2,0769	0,7086	7,9797	2,3214	0,6762	10,1899
UT	20	UNPC	2,0512	0,7120	7,7772	2,3156	0,6774	10,1310
UT	1	OSPC	4,3643	0,3692	78,5944	3,7308	0,4391	41,7125
UT	2	OSPC	3,5648	0,4560	35,3324	3,4236	0,4760	30,6797
UT	3	OSPC	3,3087	0,4875	27,3495	3,2724	0,4962	26,3746
UT	4	OSPC	3,1546	0,5070	23,4437	3,1871	0,5075	24,2181
UT	5	OSPC	3,0447	0,5212	21,0037	3,1340	0,5136	22,9657
UT	6	OSPC	2,9618	0,5321	19,3327	3,0976	0,5204	22,1447
UT	7	OSPC	2,9089	0,5410	18,3366	3,1090	0,5247	22,3986
UT	8	OSPC	2,9193	0,5475	18,5283	3,1531	0,5266	23,4085
UT	9	OSPC	2,9019	0,5547	18,2087	3,1615	0,5280	23,6060
UT	10	OSPC	2,8781	0,5618	17,7805	3,1739	0,5292	23,9005
UT	11	OSPC	2,8601	0,5681	17,4633	3,1915	0,5316	24,3249
UT	12	OSPC	2,8442	0,5738	17,1878	3,2121	0,5323	24,8312
UT	13	OSPC	2,8282	0,5794	16,9150	3,2224	0,5321	25,0883
UT	14	OSPC	2,8148	0,5842	16,6898	3,2371	0,5339	25,4598
UT	15	OSPC	2,8006	0,5886	16,4545	3,2490	0,5351	25,7646
UT	16	OSPC	2,7838	0,5930	16,1804	3,2674	0,5359	26,2430
UT	17	OSPC	2,7979	0,5915	16,4101	3,2645	0,5354	26,1670
UT	18	OSPC	2,6171	0,6238	13,6959	3,2519	0,5459	25,8394
UT	19	OSPC	2,6019	0,6313	13,4893	3,2707	0,5457	26,3298
UT	20	OSPC	2,5964	0,6346	13,4154	3,2846	0,5454	26,6983
MAMBA	1	UNPC	4,9851	0,2422	146,2182	4,0961	0,3624	60,1054
MAMBA	2	UNPC	3,902	0,3930	49,5014	3,5387	0,4593	34,4221
MAMBA	3	UNPC	3,5228	0,4522	33,8792	3,2585	0,4972	26,0105
MAMBA	4	UNPC	3,3172	0,4832	27,5830	3,1208	0,5203	22,6645

model_name	epoch	dataset	trn_loss	trn_accm	trn_pplx	val_loss	val_accm	val_pplx
MAMBA	5	UNPC	3,1834	0,5036	24,1287	2,9993	0,5390	20,0715
MAMBA	6	UNPC	3,143	0,5178	23,1733	3,1116	0,5478	22,4569
MAMBA	7	UNPC	3,2015	0,5269	24,5694	3,1151	0,5542	22,5357
MAMBA	8	UNPC	3,1725	0,5361	23,8671	3,0713	0,5642	21,5699
MAMBA	9	UNPC	3,1355	0,5449	23,0001	3,0519	0,5696	21,1555
MAMBA	10	UNPC	3,0968	0,5528	22,1270	3,0238	0,5780	20,5693
MAMBA	11	UNPC	3,0632	0,5596	21,3959	2,9817	0,5836	19,7213
MAMBA	12	UNPC	3,0337	0,5655	20,7740	2,9609	0,5875	19,3153
MAMBA	13	UNPC	3,006	0,5708	20,2064	2,9359	0,5918	18,8385
MAMBA	14	UNPC	2,9811	0,5753	19,7095	2,9293	0,5935	18,7145
MAMBA	15	UNPC	2,9627	0,5787	19,3501	2,9050	0,5969	18,2652
MAMBA	16	UNPC	2,9393	0,5827	18,9026	2,8916	0,6028	18,0221
MAMBA	17	UNPC	2,9207	0,5858	18,5543	2,8738	0,6037	17,7042
MAMBA	18	UNPC	2,9021	0,5891	18,2124	2,8654	0,6075	17,5561
MAMBA	19	UNPC	2,8848	0,5918	17,9000	2,8192	0,6103	16,7634
MAMBA	20	UNPC	2,8489	0,5957	17,2688	2,8048	0,6128	16,5238
MAMBA	1	OSPC	5,3456	0,2547	209,6837	4,6550	0,3294	105,1092
MAMBA	2	OSPC	4,4512	0,3469	85,7298	4,1807	0,3889	65,4116
MAMBA	3	OSPC	4,1371	0,3861	62,6210	3,9568	0,4172	52,2897
MAMBA	4	OSPC	3,9630	0,4085	52,6149	3,8136	0,4330	45,3133
MAMBA	5	OSPC	3,8533	0,4227	47,1484	3,7242	0,4474	41,4381
MAMBA	6	OSPC	3,7730	0,4332	43,5104	3,6644	0,4556	39,0327
MAMBA	7	OSPC	3,7103	0,4411	40,8661	3,6297	0,4602	37,7015
MAMBA	8	OSPC	3,6606	0,4477	38,8847	3,5819	0,4660	35,9418
MAMBA	9	OSPC	3,6210	0,4529	37,3749	3,5620	0,4702	35,2336
MAMBA	10	OSPC	3,5923	0,4570	36,3175	3,5613	0,4723	35,2089
MAMBA	11	OSPC	3,6066	0,4604	36,8406	3,6383	0,4757	38,0271
MAMBA	12	OSPC	3,6983	0,4621	40,3786	3,7064	0,4759	40,7070
MAMBA	13	OSPC	3,7118	0,4643	40,9274	3,6983	0,4777	40,3786
MAMBA	14	OSPC	3,7109	0,4667	40,8906	3,7045	0,4827	40,6297
MAMBA	15	OSPC	3,7080	0,4690	40,7722	3,6824	0,4829	39,7417
MAMBA	16	OSPC	3,7013	0,4718	40,4999	3,6988	0,4859	40,3988
MAMBA	17	OSPC	3,6925	0,4745	40,1451	3,6816	0,4863	39,7099
MAMBA	18	OSPC	3,6848	0,4766	39,8372	3,6884	0,4881	39,9808
MAMBA	19	OSPC	3,6734	0,4791	39,3856	3,6838	0,4906	39,7973
MAMBA	20	OSPC	3,6616	0,4815	38,9236	3,6502	0,4935	38,4824
ST-MoE	1	UNPC	4,2959	0,3501	73,3982	3,3545	0,4785	28,6313
ST-MoE	2	UNPC	3,2911	0,4875	26,8724	2,9351	0,5398	18,8234
ST-MoE	3	UNPC	2,9764	0,5314	19,6171	2,7564	0,5670	15,7431
ST-MoE	4	UNPC	2,8184	0,5566	16,7500	2,7207	0,5849	15,1910
ST-MoE	5	UNPC	2,7766	0,5752	16,0643	2,7142	0,5971	15,0925
ST-MoE	6	UNPC	2,7182	0,5919	15,1530	2,6947	0,6076	14,8011
ST-MoE	7	UNPC	2,6712	0,6052	14,4573	2,6842	0,6152	14,6465

model_name	epoch	dataset	trn_loss	trn_accm	trn_pplx	val_loss	val_accm	val_pplx
ST-MoE	8	UNPC	2,6287	0,6170	13,8557	2,6748	0,6201	14,5094
ST-MoE	9	UNPC	2,5898	0,6263	13,3271	2,6773	0,6257	14,5458
ST-MoE	10	UNPC	2,559	0,6349	12,9229	2,6553	0,6307	14,2293
ST-MoE	11	UNPC	2,5233	0,6428	12,4697	2,6658	0,6330	14,3794
ST-MoE	12	UNPC	2,4909	0,6496	12,0721	2,6624	0,6352	14,3306
ST-MoE	13	UNPC	2,4556	0,6563	11,6534	2,6446	0,6392	14,0778
ST-MoE	14	UNPC	2,422	0,6624	11,2684	2,6516	0,6396	14,1767
ST-MoE	15	UNPC	2,3817	0,6684	10,8233	2,6161	0,6428	13,6823
ST-MoE	16	UNPC	2,3369	0,6732	10,3491	2,6119	0,6454	13,6249
ST-MoE	17	UNPC	2,3117	0,6783	10,0916	2,6077	0,6464	13,5678
ST-MoE	18	UNPC	2,285	0,6831	9,8257	2,6079	0,6468	13,5705
ST-MoE	19	UNPC	2,2582	0,6881	9,5659	2,6196	0,6482	13,7302
ST-MoE	20	UNPC	2,2387	0,6924	9,3811	2,6223	0,6489	13,7674
ST-MoE	1	OSPC	4,5865	0,3512	98,1503	3,8768	0,4310	48,2695
ST-MoE	2	OSPC	3,7951	0,4384	44,4827	3,5595	0,4683	35,1456
ST-MoE	3	OSPC	3,5590	0,4681	35,1281	3,4123	0,4877	30,3349
ST-MoE	4	OSPC	3,4155	0,4863	30,4322	3,3310	0,4986	27,9663
ST-MoE	5	OSPC	3,3045	0,4997	27,2349	3,2717	0,5070	26,3561
ST-MoE	6	OSPC	3,2093	0,5102	24,7617	3,2527	0,5122	25,8601
ST-MoE	7	OSPC	3,1581	0,5194	23,5259	3,2607	0,5169	26,0678
ST-MoE	8	OSPC	3,1240	0,5278	22,7371	3,2718	0,5199	26,3587
ST-MoE	9	OSPC	3,0923	0,5361	22,0277	3,2812	0,5222	26,6077
ST-MoE	10	OSPC	3,0645	0,5440	21,4237	3,3004	0,5240	27,1235
ST-MoE	11	OSPC	3,0403	0,5505	20,9115	3,3191	0,5255	27,6355
ST-MoE	12	OSPC	3,0156	0,5578	20,4013	3,3414	0,5272	28,2587
ST-MoE	13	OSPC	2,9929	0,5640	19,9434	3,3598	0,5272	28,7834
ST-MoE	14	OSPC	2,9729	0,5700	19,5485	3,3798	0,5277	29,3649
ST-MoE	15	OSPC	2,9507	0,5761	19,1193	3,3991	0,5291	29,9371
ST-MoE	16	OSPC	2,9295	0,5817	18,7183	3,4236	0,5286	30,6797
ST-MoE	17	OSPC	2,9065	0,5870	18,2927	3,4353	0,5282	31,0407
ST-MoE	18	OSPC	2,7346	0,6212	15,4036	3,4443	0,5357	31,3214
ST-MoE	19	OSPC	2,7038	0,6301	14,9364	3,4654	0,5358	31,9893
ST-MoE	20	OSPC	2,6861	0,6345	14,6743	3,4777	0,5353	32,3852
GSS	1	UNPC	5,3583	0,1822	212,3636	5,1831	0,1978	178,2345
GSS	2	UNPC	5,2502	0,1902	190,6044	5,2712	0,1919	194,6494
GSS	3	UNPC	5,3049	0,1870	201,3209	5,1896	0,1973	179,3968
GSS	4	UNPC	5,356	0,1822	211,8757	5,7233	0,1499	305,9128
GSS	5	UNPC	5,5335	0,1681	253,0280	5,4238	0,1787	226,7391
GSS	6	UNPC	5,4146	0,1807	224,6627	5,3848	0,1906	218,0665
GSS	7	UNPC	5,4554	0,1844	234,0185	5,6001	0,1747	270,4535
GSS	8	UNPC	5,6075	0,1740	272,4622	5,5725	0,1833	263,0910
GSS	9	UNPC	5,5817	0,1802	265,5226	5,6429	0,1833	282,2801
GSS	10	UNPC	5,7314	0,1709	308,4007	5,5005	0,1976	244,8143

model_name	epoch	dataset	trn_loss	trn_accm	trn_pplx	val_loss	val_accm	val_pplx
GSS	11	UNPC	5,6188	0,1842	275,5585	5,6107	0,1910	273,3355
GSS	12	UNPC	5,6388	0,1852	281,1252	5,5900	0,1958	267,7356
GSS	13	UNPC	5,4356	0,2077	229,4305	5,4627	0,2141	235,7330
GSS	14	UNPC	5,3787	0,2138	216,7403	5,4294	0,2179	228,0124
GSS	15	UNPC	5,3656	0,2153	213,9195	5,4204	0,2199	225,9695
GSS	16	UNPC	5,3429	0,2180	209,1183	5,3983	0,2205	221,0303
GSS	17	UNPC	5,3347	0,2190	207,4105	5,3867	0,2237	218,4812
GSS	18	UNPC	5,3183	0,2212	204,0367	5,3726	0,2245	215,4222
GSS	19	UNPC	5,3168	0,2216	203,7309	5,3725	0,2250	215,4007
GSS	20	UNPC	5,3115	0,2227	202,6540	5,3566	0,2259	212,0029
GSS	1	OSPC	5,0311	0,2907	153,1013	4,5724	0,3351	96,7761
GSS	2	OSPC	4,5318	0,3352	92,9257	4,3733	0,3551	79,3049
GSS	3	OSPC	4,4670	0,3412	87,0950	4,3936	0,3516	80,9312
GSS	4	OSPC	4,4640	0,3412	86,8342	4,4812	0,3369	88,3406
GSS	5	OSPC	4,5396	0,3303	93,6533	4,3769	0,3555	79,5909
GSS	6	OSPC	4,4686	0,3392	87,2345	4,3957	0,3541	81,1014
GSS	7	OSPC	4,5182	0,3342	91,6704	4,4443	0,3465	85,1403
GSS	8	OSPC	4,4497	0,3396	85,6013	4,3386	0,3587	76,6002
GSS	9	OSPC	4,4292	0,3450	83,8643	4,3989	0,3602	81,3613
GSS	10	OSPC	4,4041	0,3519	81,7855	4,3371	0,3710	76,4854
GSS	11	OSPC	4,6041	0,3287	99,8930	4,5910	0,3412	98,5930
GSS	12	OSPC	4,6298	0,3278	102,4936	4,6070	0,3428	100,1831
GSS	13	OSPC	4,5212	0,3459	91,9459	4,5397	0,3535	93,6627
GSS	14	OSPC	4,5460	0,3446	94,2546	4,5068	0,3597	90,6313
GSS	15	OSPC	4,5034	0,3514	90,3237	4,6031	0,3459	99,7932
GSS	16	OSPC	4,4072	0,3668	82,0394	4,4589	0,3738	86,3924
GSS	17	OSPC	4,4038	0,3700	81,7610	4,5038	0,3686	90,3598
GSS	18	OSPC	4,5013	0,3577	90,1342	4,5951	0,3554	98,9980
GSS	19	OSPC	4,4905	0,3605	89,1660	4,5374	0,3680	93,4475
GSS	20	OSPC	4,5002	0,3606	90,0351	4,4256	0,3818	83,5629
MEGA	1	UNPC	4,8158	0,2583	123,4455	4,1497	0,3431	63,4150
MEGA	2	UNPC	3,8622	0,3892	47,5699	3,4998	0,4515	33,1088
MEGA	3	UNPC	3,3744	0,4702	29,2068	3,1529	0,5116	23,4038
MEGA	4	UNPC	3,0774	0,5192	21,7019	2,9336	0,5462	18,7952
MEGA	5	UNPC	2,895	0,5500	18,0835	2,8114	0,5712	16,6332
MEGA	6	UNPC	2,8639	0,5718	17,5298	2,8948	0,5877	18,0799
MEGA	7	UNPC	2,8636	0,5886	17,5245	2,8652	0,5977	17,5526
MEGA	8	UNPC	2,8021	0,6022	16,4792	2,8156	0,6070	16,7032
MEGA	9	UNPC	2,749	0,6135	15,6270	2,7861	0,6158	16,2176
MEGA	10	UNPC	2,7027	0,6230	14,9200	2,7445	0,6238	15,5568
MEGA	11	UNPC	2,6601	0,6315	14,2977	2,7182	0,6301	15,1530
MEGA	12	UNPC	2,6193	0,6386	13,7261	2,6917	0,6344	14,7567
MEGA	13	UNPC	2,591	0,6444	13,3431	2,6886	0,6361	14,7111

model_name	epoch	dataset	trn_loss	trn_accm	trn_pplx	val_loss	val_accm	val_pplx
MEGA	14	UNPC	2,5549	0,6505	12,8700	2,6603	0,6412	14,3006
MEGA	15	UNPC	2,523	0,6555	12,4659	2,6449	0,6438	14,0820
MEGA	16	UNPC	2,4937	0,6601	12,1060	2,6301	0,6456	13,8752
MEGA	17	UNPC	2,4647	0,6644	11,7600	2,6189	0,6484	13,7206
MEGA	18	UNPC	2,4218	0,6688	11,2661	2,5656	0,6501	13,0085
MEGA	19	UNPC	2,3892	0,6717	10,9048	2,5876	0,6501	13,2978
MEGA	20	UNPC	2,3734	0,6749	10,7338	2,5633	0,6520	12,9786
MEGA	1	OSPC	4,9958	0,2941	147,7911	4,3568	0,3633	78,0071
MEGA	2	OSPC	4,1487	0,3883	63,3516	3,9059	0,4238	49,6948
MEGA	3	OSPC	3,8175	0,4323	45,4903	3,7008	0,4528	40,4797
MEGA	4	OSPC	3,6144	0,4589	37,1291	3,5426	0,4716	34,5566
MEGA	5	OSPC	3,4730	0,4770	32,2333	3,4495	0,4847	31,4846
MEGA	6	OSPC	3,3691	0,4903	29,0524	3,3738	0,4944	29,1892
MEGA	7	OSPC	3,2902	0,5007	26,8482	3,3275	0,5024	27,8686
MEGA	8	OSPC	3,2740	0,5092	26,4168	3,3645	0,5080	28,9190
MEGA	9	OSPC	3,2571	0,5173	25,9741	3,3613	0,5120	28,8266
MEGA	10	OSPC	3,2330	0,5242	25,3556	3,3673	0,5147	29,0001
MEGA	11	OSPC	3,2141	0,5300	24,8809	3,3654	0,5197	28,9451
MEGA	12	OSPC	3,1975	0,5356	24,4713	3,3709	0,5208	29,1047
MEGA	13	OSPC	3,1840	0,5405	24,1431	3,3826	0,5227	29,4472
MEGA	14	OSPC	3,1706	0,5449	23,8218	3,3840	0,5256	29,4885
MEGA	15	OSPC	3,1568	0,5493	23,4953	3,3911	0,5267	29,6986
MEGA	16	OSPC	3,1425	0,5535	23,1617	3,3915	0,5280	29,7105
MEGA	17	OSPC	3,1281	0,5573	22,8306	3,4052	0,5290	30,1203
MEGA	18	OSPC	3,1143	0,5612	22,5177	3,4073	0,5300	30,1836
MEGA	19	OSPC	2,9630	0,5873	19,3560	3,3726	0,5413	29,1542
MEGA	20	OSPC	2,9454	0,5933	19,0183	3,3814	0,5415	29,4119

Anexo No. 3. Uso de memoria y duración por épocas

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
Vainilla	1	UNPC	0,0329	562,9676	3252,4778	0,0324	560,9840	493,1936
Vainilla	2	UNPC	0,0330	564,6744	3117,4205	0,0324	560,7176	463,9956
Vainilla	3	UNPC	0,0329	563,0196	3082,9082	0,0323	559,3332	442,6457
Vainilla	4	UNPC	0,0329	563,0452	3049,8695	0,0324	561,0144	448,5050
Vainilla	5	UNPC	0,0328	562,7080	3218,9603	0,0323	559,4984	492,9527
Vainilla	6	UNPC	0,0328	562,3984	3115,2957	0,0323	559,4900	485,2851
Vainilla	7	UNPC	0,0329	562,9380	3145,7566	0,0323	559,8408	499,6470
Vainilla	8	UNPC	0,0329	563,0524	3137,8544	0,0324	560,8108	474,9537
Vainilla	9	UNPC	0,0329	563,4404	3177,3929	0,0323	559,5912	499,4537
Vainilla	10	UNPC	0,0328	562,5828	3203,6590	0,0323	560,2000	504,5685
Vainilla	11	UNPC	0,0329	563,7148	3213,9510	0,0322	559,0952	480,2847
Vainilla	12	UNPC	0,0329	563,0232	3166,0436	0,0324	560,5588	484,3070
Vainilla	13	UNPC	0,0329	563,4952	3147,7708	0,0323	559,4656	485,2911
Vainilla	14	UNPC	0,0329	563,5236	3173,8746	0,0324	559,9536	458,9870
Vainilla	15	UNPC	0,0329	563,0872	3088,6795	0,0323	560,0592	475,5751
Vainilla	16	UNPC	0,0329	563,7664	3083,7847	0,0324	561,1404	466,5793
Vainilla	17	UNPC	0,0329	562,9472	3090,4371	0,0324	560,8564	467,4164
Vainilla	18	UNPC	0,0329	563,4232	3091,9812	0,0322	558,7724	470,3111
Vainilla	19	UNPC	0,0329	563,4308	3177,4965	0,0323	559,5284	513,5908
Vainilla	20	UNPC	0,0329	563,1668	3242,8158	0,0324	560,2820	506,0144
Vainilla	1	OSPC	0,0311	531,7691	3829,0898	0,0311	531,3675	614,3776
Vainilla	2	OSPC	0,0311	531,0739	3817,8935	0,0312	532,3107	567,0638
Vainilla	3	OSPC	0,0311	531,9374	3712,0179	0,0312	533,1308	580,5806
Vainilla	4	OSPC	0,0311	531,4155	3917,7411	0,0312	533,5851	609,2572
Vainilla	5	OSPC	0,0311	530,7698	3806,1444	0,0312	532,9012	595,4835
Vainilla	6	OSPC	0,0311	531,7691	3801,9382	0,0311	531,3675	588,7136
Vainilla	7	OSPC	0,0311	531,0739	3923,6787	0,0312	532,3107	612,6289
Vainilla	8	OSPC	0,0311	531,9374	3824,6278	0,0312	533,1308	608,8780
Vainilla	9	OSPC	0,0311	531,4155	3863,4443	0,0312	533,5851	628,4010
Vainilla	10	OSPC	0,0311	530,7698	3815,0321	0,0312	532,9012	590,5316
Vainilla	11	OSPC	0,0311	531,1516	3815,6334	0,0312	533,7038	598,0743
Vainilla	12	OSPC	0,0311	531,5566	3822,5413	0,0311	531,7988	591,0423
Vainilla	13	OSPC	0,0311	531,7573	3760,1205	0,0312	532,8547	590,9767
Vainilla	14	OSPC	0,0310	530,7907	3842,4261	0,0311	531,7194	585,8434
Vainilla	15	OSPC	0,0311	531,7691	3948,7794	0,0311	531,3675	595,2735
Vainilla	16	OSPC	0,0311	531,0739	4019,1894	0,0312	532,3107	588,7569

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
Vainilla	17	OSPC	0,0311	531,9374	3972,2276	0,0312	533,1308	568,5948
Vainilla	18	OSPC	0,0311	531,4155	3882,6909	0,0312	533,5851	529,9251
Vainilla	19	OSPC	0,0311	530,7698	3820,7628	0,0312	532,9012	516,5415
Vainilla	20	OSPC	0,0311	531,1516	3656,9288	0,0312	533,7038	541,0311
UT	1	UNPC	0,0080	562,6716	4792,7157	0,0080	560,0728	668,9942
UT	2	UNPC	0,0080	564,0468	4950,8178	0,0080	559,0100	653,2644
UT	3	UNPC	0,0080	564,2448	4813,2657	0,0080	559,7924	645,8403
UT	4	UNPC	0,0080	563,5120	4868,2482	0,0080	558,3280	675,8555
UT	5	UNPC	0,0080	563,6140	4908,4884	0,0080	559,8724	678,9933
UT	6	UNPC	0,0080	561,7948	4782,3002	0,0080	560,7828	649,5450
UT	7	UNPC	0,0080	563,9608	4876,9516	0,0080	560,3084	661,1203
UT	8	UNPC	0,0080	563,5088	4762,5583	0,0080	559,8716	638,5373
UT	9	UNPC	0,0080	563,1716	4935,8223	0,0080	559,7356	669,2653
UT	10	UNPC	0,0080	563,6212	4923,7648	0,0080	560,2288	675,7470
UT	11	UNPC	0,0080	563,2892	5044,4528	0,0080	560,4576	714,8989
UT	12	UNPC	0,0080	562,5196	4934,5474	0,0080	559,6692	685,4597
UT	13	UNPC	0,0080	563,6064	4923,3137	0,0080	558,6956	691,9752
UT	14	UNPC	0,0080	563,1884	4918,8142	0,0080	558,9192	688,1158
UT	15	UNPC	0,0080	562,9880	4930,6052	0,0080	560,6948	680,8276
UT	16	UNPC	0,0080	563,4844	4964,4458	0,0080	560,3332	685,9692
UT	17	UNPC	0,0080	562,9640	4940,6114	0,0080	560,6216	697,9747
UT	18	UNPC	0,0080	563,3560	4987,5865	0,0080	560,9100	688,2655
UT	19	UNPC	0,0080	563,0912	5012,3937	0,0080	559,7892	687,4299
UT	20	UNPC	0,0080	563,4880	4933,1812	0,0080	560,1120	675,8899
UT	1	OSPC	0,0249	531,4151	6012,8160	0,0250	534,3263	713,7601
UT	2	OSPC	0,0249	530,5014	6090,1927	0,0249	531,7447	771,0414
UT	3	OSPC	0,0249	531,1959	6094,2643	0,0250	533,7884	817,3785
UT	4	OSPC	0,0249	530,7951	6107,0390	0,0249	532,8755	779,7242
UT	5	OSPC	0,0248	530,4673	5951,1427	0,0249	530,4857	764,2385
UT	6	OSPC	0,0249	531,7495	5914,8786	0,0249	533,3267	741,0186
UT	7	OSPC	0,0249	531,1245	5926,3239	0,0249	532,3832	780,4987
UT	8	OSPC	0,0249	531,5396	6171,7750	0,0249	532,4725	745,8006
UT	9	OSPC	0,0249	531,4151	6685,3406	0,0250	534,3263	838,3370
UT	10	OSPC	0,0249	530,5014	6895,6878	0,0249	531,7447	807,1089
UT	11	OSPC	0,0249	531,1959	6438,3250	0,0250	533,7884	829,0382
UT	12	OSPC	0,0249	530,7951	6543,4687	0,0249	532,8755	783,7475
UT	13	OSPC	0,0248	530,4673	6518,7394	0,0249	530,4857	793,2338

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
UT	14	OSPC	0,0249	531,7495	6489,6098	0,0249	533,3267	722,4051
UT	15	OSPC	0,0249	531,1245	6118,1254	0,0249	532,3832	727,7372
UT	16	OSPC	0,0249	531,5396	6080,2145	0,0249	532,4725	743,8036
UT	17	OSPC	0,0249	531,4197	6041,0588	0,0250	533,5431	743,4146
UT	18	OSPC	0,0249	531,6421	6092,6846	0,0250	533,7092	760,7311
UT	19	OSPC	0,0249	531,1417	6593,3507	0,0250	533,5304	836,1146
UT	20	OSPC	0,0249	531,8941	6993,0552	0,0249	533,3428	881,8909
MAMBA	1	UNPC	0,0080	563,0208	3402,5075	0,0080	558,7344	592,0435
MAMBA	2	UNPC	0,0080	562,5592	3425,9119	0,0080	558,8116	593,5259
MAMBA	3	UNPC	0,0080	562,9780	3480,9590	0,0080	559,0172	598,7798
MAMBA	4	UNPC	0,0080	563,5708	3481,9169	0,0080	560,4560	623,4210
MAMBA	5	UNPC	0,0080	563,3772	3504,5645	0,0080	558,6772	624,5111
MAMBA	6	UNPC	0,0080	562,9520	3488,9977	0,0080	558,1880	604,8754
MAMBA	7	UNPC	0,0080	563,9532	3507,7827	0,0080	560,3564	624,4096
MAMBA	8	UNPC	0,0080	563,3164	3562,3346	0,0080	562,0328	618,8800
MAMBA	9	UNPC	0,0080	564,0652	3563,2849	0,0080	559,2616	635,9067
MAMBA	10	UNPC	0,0080	562,9264	3667,3007	0,0080	559,8836	615,2520
MAMBA	11	UNPC	0,0080	563,0688	3609,8970	0,0080	559,9464	641,6283
MAMBA	12	UNPC	0,0080	563,0668	3605,0222	0,0080	561,6520	622,7757
MAMBA	13	UNPC	0,0080	563,8268	3473,4220	0,0080	558,9872	580,4235
MAMBA	14	UNPC	0,0080	564,4260	3616,7546	0,0080	558,0760	621,9412
MAMBA	15	UNPC	0,0080	563,1880	3467,9522	0,0080	559,8752	604,3434
MAMBA	16	UNPC	0,0080	562,6200	3472,9991	0,0080	559,4412	609,6284
MAMBA	17	UNPC	0,0080	562,6992	3465,7592	0,0080	560,8748	608,7233
MAMBA	18	UNPC	0,0080	563,8612	3475,1991	0,0080	559,5064	606,0925
MAMBA	19	UNPC	0,0080	563,1148	3477,5204	0,0080	560,6876	612,3055
MAMBA	20	UNPC	0,0080	562,3324	3466,9826	0,0080	560,0420	616,1288
MAMBA	1	OSPC	0,0249	530,9372	3757,0523	0,0249	532,0373	639,0693
MAMBA	2	OSPC	0,0249	531,4362	3697,2371	0,0250	533,5681	575,2323
MAMBA	3	OSPC	0,0249	531,2312	3923,3709	0,0249	531,6469	668,0610
MAMBA	4	OSPC	0,0249	531,1579	3720,9249	0,0249	531,6680	647,5013
MAMBA	5	OSPC	0,0248	530,4386	3832,9272	0,0249	532,3881	643,7462
MAMBA	6	OSPC	0,0249	531,3418	3797,0692	0,0250	533,3362	643,6227
MAMBA	7	OSPC	0,0249	531,6108	3804,8359	0,0249	532,5275	656,2468
MAMBA	8	OSPC	0,0249	531,7544	3574,6195	0,0250	535,2073	601,5565
MAMBA	9	OSPC	0,0249	531,1753	3612,5586	0,0250	532,8009	616,6785
MAMBA	10	OSPC	0,0249	530,5099	3637,4093	0,0249	533,0371	617,9368

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
MAMBA	11	OSPC	0,0249	530,9641	3819,8462	0,0249	533,1834	649,9704
MAMBA	12	OSPC	0,0249	530,9431	3657,8597	0,0248	530,6900	638,8118
MAMBA	13	OSPC	0,0249	530,7598	3687,4639	0,0250	533,9357	641,8932
MAMBA	14	OSPC	0,0249	531,2590	3612,1831	0,0250	533,9849	608,9789
MAMBA	15	OSPC	0,0249	532,1212	3704,4967	0,0249	531,3292	651,5238
MAMBA	16	OSPC	0,0249	531,7498	3891,3037	0,0249	532,7206	610,5989
MAMBA	17	OSPC	0,0249	531,5112	3623,1413	0,0250	534,3151	620,8768
MAMBA	18	OSPC	0,0249	530,8338	3821,2938	0,0249	532,2156	614,4388
MAMBA	19	OSPC	0,0249	531,2951	3848,0875	0,0249	532,2052	652,9046
MAMBA	20	OSPC	0,0249	531,3377	3838,0462	0,0249	531,1324	664,9477
ST-MoE	1	UNPC	0,0080	563,6644	3871,5407	0,0080	558,8364	582,5655
ST-MoE	2	UNPC	0,0080	563,4148	3924,7202	0,0080	559,3008	635,8916
ST-MoE	3	UNPC	0,0080	562,6604	4093,5019	0,0080	558,6240	642,7858
ST-MoE	4	UNPC	0,0080	562,9768	4025,8507	0,0080	557,9900	638,9806
ST-MoE	5	UNPC	0,0080	563,6360	4039,0190	0,0080	558,5984	616,9055
ST-MoE	6	UNPC	0,0080	563,1872	4043,5296	0,0079	557,5504	625,6697
ST-MoE	7	UNPC	0,0080	563,2648	4042,4262	0,0080	560,0132	650,0579
ST-MoE	8	UNPC	0,0080	563,2352	4057,2061	0,0080	559,2196	629,7170
ST-MoE	9	UNPC	0,0080	563,3224	4080,5789	0,0080	560,9620	629,5483
ST-MoE	10	UNPC	0,0080	563,6032	4019,0397	0,0080	558,8236	632,2203
ST-MoE	11	UNPC	0,0080	563,6740	4091,0646	0,0080	560,0976	664,2030
ST-MoE	12	UNPC	0,0080	562,8500	4043,7053	0,0080	560,7120	629,0457
ST-MoE	13	UNPC	0,0080	563,0992	4108,6284	0,0080	557,8588	678,2561
ST-MoE	14	UNPC	0,0080	563,5156	4148,1885	0,0080	560,5224	626,0521
ST-MoE	15	UNPC	0,0080	563,3188	4067,9162	0,0080	558,3188	644,3566
ST-MoE	16	UNPC	0,0080	563,0644	4087,3120	0,0080	560,1284	658,6393
ST-MoE	17	UNPC	0,0080	563,7832	4110,7134	0,0080	559,7180	656,0169
ST-MoE	18	UNPC	0,0080	564,0984	4187,7633	0,0080	561,6252	675,6108
ST-MoE	19	UNPC	0,0080	563,2692	4280,4183	0,0080	558,5280	650,2625
ST-MoE	20	UNPC	0,0080	563,0944	4123,2744	0,0080	559,0820	667,9472
ST-MoE	1	OSPC	0,0249	531,3418	4550,6574	0,0250	532,9778	715,5589
ST-MoE	2	OSPC	0,0249	531,4695	4697,8988	0,0249	532,7380	701,6707
ST-MoE	3	OSPC	0,0249	531,1831	4567,6808	0,0249	532,3044	700,0657
ST-MoE	4	OSPC	0,0249	531,9801	4411,6314	0,0250	533,4114	668,2974
ST-MoE	5	OSPC	0,0249	531,7376	4587,9481	0,0249	532,6193	708,1485
ST-MoE	6	OSPC	0,0249	531,2929	4763,6342	0,0249	532,1746	669,8132
ST-MoE	7	OSPC	0,0249	531,3190	4323,5320	0,0250	534,2979	674,7554

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
ST-MoE	8	OSPC	0,0249	531,7283	4357,6150	0,0250	532,6207	677,3291
ST-MoE	9	OSPC	0,0249	531,9843	4367,1127	0,0249	533,3899	697,6170
ST-MoE	10	OSPC	0,0248	530,7625	4400,4749	0,0249	532,0852	676,1899
ST-MoE	11	OSPC	0,0249	531,3247	4382,7822	0,0249	532,5955	682,4076
ST-MoE	12	OSPC	0,0249	532,0633	4464,5964	0,0250	535,5409	682,6444
ST-MoE	13	OSPC	0,0249	530,8625	4449,7406	0,0249	532,6255	692,3766
ST-MoE	14	OSPC	0,0249	531,7837	5028,9436	0,0250	534,2120	742,2194
ST-MoE	15	OSPC	0,0249	532,0773	5101,2794	0,0249	532,0971	748,2042
ST-MoE	16	OSPC	0,0249	530,7235	4972,1925	0,0249	530,4925	805,7025
ST-MoE	17	OSPC	0,0248	530,7734	5144,0237	0,0250	533,2409	732,4290
ST-MoE	18	OSPC	0,0249	531,4938	5294,3182	0,0250	533,9749	711,2659
ST-MoE	19	OSPC	0,0249	531,1947	5432,7681	0,0250	533,1780	854,4739
ST-MoE	20	OSPC	0,0249	531,2571	5572,6693	0,0249	532,5117	701,2803
GSS	1	UNPC	0,0080	563,1028	4205,9107	0,0080	559,0632	650,0726
GSS	2	UNPC	0,0080	563,2096	4077,7127	0,0080	560,8636	668,0589
GSS	3	UNPC	0,0080	562,9528	4089,3136	0,0080	560,1360	671,8298
GSS	4	UNPC	0,0080	563,5604	4131,7285	0,0080	559,6100	666,7918
GSS	5	UNPC	0,0080	563,4640	4094,7222	0,0080	559,3588	667,4561
GSS	6	UNPC	0,0080	562,9784	4091,7376	0,0080	559,2128	668,2887
GSS	7	UNPC	0,0080	564,4380	4092,8928	0,0080	559,9856	673,9598
GSS	8	UNPC	0,0080	562,7592	4092,4580	0,0080	559,6700	672,1534
GSS	9	UNPC	0,0080	562,9112	4100,8393	0,0080	560,4144	671,3768
GSS	10	UNPC	0,0080	563,1936	4095,1755	0,0080	559,8628	674,3078
GSS	11	UNPC	0,0080	563,6288	4106,5804	0,0080	560,0276	679,8970
GSS	12	UNPC	0,0080	563,4332	4110,8664	0,0080	561,0056	679,0814
GSS	13	UNPC	0,0080	563,2216	4071,0252	0,0080	559,8492	645,5564
GSS	14	UNPC	0,0080	562,4368	3980,4288	0,0080	560,5876	636,8885
GSS	15	UNPC	0,0080	562,5772	3989,7846	0,0080	560,2944	635,9740
GSS	16	UNPC	0,0080	563,1420	4106,4023	0,0080	561,8488	697,4448
GSS	17	UNPC	0,0080	563,3732	4191,1865	0,0080	559,7076	689,2295
GSS	18	UNPC	0,0080	562,6704	4131,8163	0,0080	561,4356	687,9272
GSS	19	UNPC	0,0080	563,5900	4304,9100	0,0080	559,3852	691,8394
GSS	20	UNPC	0,0080	563,4032	4165,1359	0,0080	560,1648	690,5499
GSS	1	OSPC	0,0249	531,6434	4932,8126	0,0250	533,0175	786,4483
GSS	2	OSPC	0,0249	530,4937	4949,8462	0,0250	532,2382	800,7982
GSS	3	OSPC	0,0249	531,7898	4890,2161	0,0250	534,2915	796,7790
GSS	4	OSPC	0,0249	531,4183	4973,4269	0,0249	532,4866	806,1134

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
GSS	5	OSPC	0,0249	530,9528	5213,5441	0,0250	531,9312	956,1572
GSS	6	OSPC	0,0249	530,9461	7416,4301	0,0249	532,7412	1209,7037
GSS	7	OSPC	0,0249	531,6434	4260,0176	0,0250	533,0175	590,8957
GSS	8	OSPC	0,0249	530,4937	4400,2753	0,0250	532,2382	678,8286
GSS	9	OSPC	0,0249	531,7898	4276,7809	0,0250	534,2915	622,2903
GSS	10	OSPC	0,0249	531,4183	4324,1457	0,0249	532,4866	607,3698
GSS	11	OSPC	0,0249	530,9528	4123,5411	0,0250	531,9312	645,1791
GSS	12	OSPC	0,0249	530,9461	4060,5523	0,0249	532,7412	625,9633
GSS	13	OSPC	0,0249	531,3044	4284,3108	0,0250	533,2383	625,4664
GSS	14	OSPC	0,0249	530,7478	4109,8520	0,0250	532,6291	660,4261
GSS	15	OSPC	0,0249	531,8455	4205,7913	0,0250	533,7343	626,9023
GSS	16	OSPC	0,0249	531,7278	4259,1337	0,0250	533,5230	601,0579
GSS	17	OSPC	0,0248	530,9554	3978,5107	0,0249	533,7785	589,0557
GSS	18	OSPC	0,0249	531,1156	3972,1269	0,0250	534,0071	594,4537
GSS	19	OSPC	0,0249	530,9893	3977,6146	0,0249	533,5100	576,9683
GSS	20	OSPC	0,0248	531,3348	3958,2560	0,0249	532,9011	593,4256
MEGA	1	UNPC	0,0080	563,0404	5283,4472	0,0080	560,5632	757,7263
MEGA	2	UNPC	0,0080	562,7300	5325,3194	0,0080	560,0084	767,3134
MEGA	3	UNPC	0,0080	563,5588	5352,5379	0,0080	559,1364	796,8020
MEGA	4	UNPC	0,0080	562,5208	5298,3803	0,0080	559,6808	802,2413
MEGA	5	UNPC	0,0080	563,1856	5290,6898	0,0080	561,6480	773,2582
MEGA	6	UNPC	0,0080	563,2648	5377,9656	0,0080	559,3780	774,0481
MEGA	7	UNPC	0,0080	563,5140	5382,9373	0,0080	559,2924	807,9502
MEGA	8	UNPC	0,0080	564,1488	5309,2461	0,0080	558,7140	756,9659
MEGA	9	UNPC	0,0080	563,2044	5312,0664	0,0080	559,5484	774,8150
MEGA	10	UNPC	0,0080	562,9704	5190,7091	0,0080	559,8600	755,6971
MEGA	11	UNPC	0,0080	563,4752	5092,7188	0,0080	559,3804	761,3720
MEGA	12	UNPC	0,0080	563,6768	5097,0430	0,0080	560,9356	756,3194
MEGA	13	UNPC	0,0080	563,3712	5143,5562	0,0080	559,4868	762,2020
MEGA	14	UNPC	0,0080	563,0184	5104,4121	0,0080	559,5692	743,9114
MEGA	15	UNPC	0,0080	563,3952	5197,8677	0,0080	558,7588	756,7967
MEGA	16	UNPC	0,0080	563,2836	5219,1782	0,0080	559,6936	763,0206
MEGA	17	UNPC	0,0080	564,1564	5211,5501	0,0080	559,3752	752,0593
MEGA	18	UNPC	0,0080	563,7104	5180,6254	0,0080	559,3464	809,4031
MEGA	19	UNPC	0,0080	563,3368	5384,2676	0,0080	561,4504	787,3313
MEGA	20	UNPC	0,0080	562,8280	5173,9321	0,0080	560,6512	751,5044
MEGA	1	OSPC	0,0249	531,6094	6995,7230	0,0249	532,1452	855,2217

Model	Epoch	Dataset	trn_wrkmem	trn_logmem	trn_elapsed_time	val_wrkmem	val_logmem	val_elapsed_time
MEGA	2	OSPC	0,0249	531,3593	7525,4801	0,0250	533,3827	802,8953
MEGA	3	OSPC	0,0249	531,4142	7894,7118	0,0250	533,8706	852,1953
MEGA	4	OSPC	0,0249	531,6456	7700,8033	0,0250	534,2156	878,8472
MEGA	5	OSPC	0,0248	530,5109	7387,8470	0,0249	531,0159	789,1979
MEGA	6	OSPC	0,0249	531,2395	7444,7669	0,0250	533,6896	816,0129
MEGA	7	OSPC	0,0249	531,1177	6830,7051	0,0249	532,8838	810,4687
MEGA	8	OSPC	0,0249	530,9699	6936,3907	0,0249	532,0157	814,4279
MEGA	9	OSPC	0,0249	531,7216	6796,7657	0,0249	533,0030	825,8077
MEGA	10	OSPC	0,0248	530,7437	6901,3598	0,0249	532,8084	830,8458
MEGA	11	OSPC	0,0249	531,7943	6938,7401	0,0250	533,9897	825,8955
MEGA	12	OSPC	0,0249	532,1199	7474,2026	0,0249	533,4410	916,0110
MEGA	13	OSPC	0,0249	531,6173	7454,2906	0,0249	531,7266	893,3805
MEGA	14	OSPC	0,0249	531,3318	9066,3482	0,0249	532,8614	966,2570
MEGA	15	OSPC	0,0249	531,6094	8042,2644	0,0249	532,1452	834,7902
MEGA	16	OSPC	0,0249	531,3593	8041,1181	0,0250	533,3827	820,9235
MEGA	17	OSPC	0,0249	531,4142	7712,5990	0,0250	533,8706	813,0349
MEGA	18	OSPC	0,0249	531,6456	7406,1757	0,0250	534,2156	823,2414
MEGA	19	OSPC	0,0248	530,5109	7226,6300	0,0249	531,0159	802,9231
MEGA	20	OSPC	0,0249	531,2395	7465,1655	0,0250	533,6896	845,7373