
Universidad Politécnica
de Madrid

**Escuela Técnica Superior de Ingenieros Informáticos**


Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

# Experimentos con Temporal Graph Networks


Autor(a): Javier Rodríguez Sánchez
Tutor(a): Damiano Zanardini


Madrid, Julio 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Máster*
*Máster Universitario en* Inteligencia Artificial

*Título:* Experimentos con Temporal Graph Networks

Julio 2024

*Autor(a):* Javier Rodríguez Sánchez
*Tutor(a):* Damiano Zanardini
Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

# Agradecimientos

Me gustaría agradecer el apoyo que he tenido durante la realización de este trabajo por parte de todas las personas cercanas a mi. Para comenzar, a mis padres que siempre han estado para apoyarme en mis estudios, haciendo que nunca sea suficiente y siempre quiera superarme. También se lo agradezco a mis amigos, los cuales me han ayudado a desconectar en momentos en los que lo necesitaba para volver con más fuerzas. Por último, me gustaría agradecérselo a Lucía, la cual ha estado este año a mi lado siempre haciéndome sentir la persona más capacitada del mundo para conseguir terminar este trabajo y el máster de Inteligencia Artificial.

# Resumen

Partiendo de la base establecida en el artículo Rossi et al. (2020) y aprovechando el repositorio correspondiente disponible en https://github.com/twitterresearch/tgn, este trabajo consiste en una evaluación experimental exhaustiva del marco de trabajo de la Red de Grafos Temporales (TGN) tal y como se delinea en el estudio referenciado. Esta evaluación es polifacética e incluye no sólo la repetición y el análisis de los experimentos originales, sino también un examen crítico de las futuras líneas de investigación propuestas en este.

Como introducción, se ofrece una visión general de los grafos. Esto incluye un análisis detallado de los distintos tipos de grafos, los problemas de predicción asociados a ellos y las metodologías empleadas para resolverlos. Esta visión general pretende dotar al lector de un conocimiento profundo de los conceptos y retos fundamentales de la teoría de grafos, así como de las estrategias empleadas para resolver los problemas de predicción relacionados.

Para completar la contextualización de este trabajo, se dedica un capítulo a los trabajos previos. Este capítulo explora las primeras aplicaciones de los métodos de aprendizaje automático, en particular las redes neuronales, en el contexto del trabajo con grafos. Presenta una visión general de algunos de los artículos más influyentes y de las soluciones comúnmente adoptadas en este ámbito, proporcionando una perspectiva histórica y una comprensión fundacional de los avances que han dado forma a las metodologías actuales como la que es objeto a estudio en este trabajo.

Una vez contextualizada la aplicación de las redes neuronales a las estructuras de grafos, se presenta en detalle el funcionamiento del modelo TGN propuesto por Rossi et al. (2020). Este capítulo incluye una explicación exhaustiva del flujo operativo del modelo, sus diversos módulos, los métodos de entrenamiento empleados y los conjuntos de datos utilizados. Esta presentación detallada tiene por objeto proporcionar una comprensión completa del modelo TGN, destacando su diseño, funcionalidad y las consideraciones prácticas que implica su aplicación.

Además de este análisis fundacional, el trabajo se extiende para incorporar nuevas vías de investigación que han surgido en la literatura más reciente relacionada con las redes de grafos temporales. Al sintetizar las ideas de estos estudios contemporáneos, pretendemos identificar posibles metodologías que podrían mejorar aún más el rendimiento y la aplicabilidad de la TGN.

El alcance del estudio incluye una investigación detallada de varias modificaciones arquitectónicas, técnicas de optimización y metodologías alternativas para la selección temporal de vecinos. Mediante pruebas rigurosas de estos diferentes enfoques, se evaluará su impacto en la eficiencia, precisión y velocidad de cálculo del modelo.

# Abstract

Building on the foundation established in the article Rossi et al. (2020) and taking advantage of the corresponding repository available at https://github.com/twitterresearch/tgn, this work consists of a comprehensive experimental evaluation of the Temporal Graph Network (TGN) framework as delineated in the referenced study. This evaluation is multifaceted and includes not only the repetition and analysis of the original experiments, but also a critical examination of the future lines of research proposed in this one.

As an introduction to this paper, an overview of networks is given. This includes a detailed analysis of the different types of graphs, the prediction problems associated with them and the methodologies used to solve them. This overview is intended to provide the reader with a thorough understanding of the fundamental concepts and challenges of graph theory, as well as the strategies employed to solve the related prediction problems.

To complete the contextualization of this work, a chapter is devoted to previous work. This chapter explores early applications of machine learning methods, in particular neural networks, in the context of working with graphs. It presents an overview of some of the most influential papers and commonly adopted solutions in this area, providing a historical perspective and a foundational understanding of the advances that have shaped current methodologies such as the one under study in this paper.

Once the application of neural networks to graph structures has been contextualized, the operation of the TGN model proposed by Rossi et al. (2020) is presented in detail. This chapter includes a comprehensive explanation of the operational flow of the model, its various modules, the training methods employed and the data sets used. This detailed presentation is intended to provide a thorough understanding of the TGN model, highlighting its design, functionality, and the practical considerations involved in its application.

In addition to this foundational analysis, the paper extends to incorporate new avenues of research that have emerged in the more recent literature related to temporal graph networks. By synthesizing insights from these contemporary studies, we aim to identify possible methodologies that could further enhance the performance and applicability of TGN.

The scope of this study includes a detailed investigation of various architectural modifications, optimization techniques, and alternative methodologies for temporal neighbor selection. Through rigorous testing of these different approaches, their impact on model efficiency, accuracy and computational speed will be evaluated.

# Tabla de contenidos

# Capítulo 1

# Introducción

Durante este capítulo introductorio, se abordarán los conceptos fundamentales de los grafos, presentando una descripción detallada de dos tipos principales, grafos estáticos y dinámicos junto con su notación. Las propiedades de cada tipo de grafo y la notación matemática serán expuestas para proporcionar una base sólida para la comprensión de este trabajo.

Además, se examinarán los tres problemas clave de predicción en grafos, la predicción de enlaces, la clasificación de nodos y la predicción de propiedades de grafos. Entre los métodos para abordar estos problemas, se introducirán dos de ellos, que son los más importantes en el área a tratar; el uso de redes neuronales orientadas a grafos y el aprendizaje de representación de grafos.

Por último, se explicarán los dos tipos de entorno de aprendizaje, transductivo e inductivo, distinción la cuál es crucial para entender cómo se aplican las técnicas de redes neuronales y el aprendizaje de representaciones en diferentes contextos y escenarios de predicción en grafos.

## 1.1. Conceptos básicos sobre los grafos y notación

Para abordar con rigurosidad el análisis del artículo Rossi et al. (2020) y entender los cambios propuestos sobre el modelo en este trabajo, es imperativo situarse en el contexto de los grafos, así como comprender los diversos tipos que conforman esta estructura fundamental en teoría de grafos y matemáticas discretas.

### 1.1.1. Grafos

Un grafo es una estructura matemática que representa relaciones entre objetos. Consiste en un conjunto de nodos o vértices que están conectados por aristas o bordes. Los grafos son utilizados para modelar una variedad de situaciones en el mundo real donde existen conexiones o relaciones entre elementos.

Los grafos son indispensables por su capacidad para representar relaciones entre entidades, lo que los hace versátiles para modelar y analizar estructuras y dependencias en diversos campos. Destacan por representar conexiones entre nodos, que pueden simbolizar entidades como personas, ciudades o productos, mientras que las aristas significan interacciones entre ellos. Esta representación es fundamental en

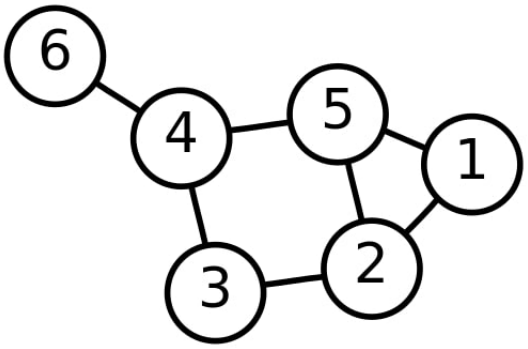



Figura 1.1: Grafo etiquetado con 6 vértices y 7 aristas

campos como las redes sociales, los sistemas de transporte y la gestión de la cadena de suministro.

Además, los grafos desempeñan un papel crucial en el análisis de redes, ya que proporcionan información sobre patrones, agrupaciones y nodos influyentes en redes sociales, redes de comunicación y redes biológicas. Por otra parte, también son esenciales para los problemas de optimización, como la búsqueda del camino más corto en redes de transporte o la optimización de rutas para servicios de reparto.

Por último, la capacidad de los grafos para modelar relaciones temporales y espaciales amplía aún más su aplicabilidad, permitiendo a los investigadores y profesionales abordar problemas complejos y dinámicos en tiempo real.

A continuación y debido a que el artículo Rossi et al. (2020) trata sobre uno de estos, se explicarán dos tipos importantes de grafos, los grafos estáticos y los grafos dinámicos. Cabe destacar que existen otras formas de clasificarlos, tales como si son dirigidos o no dirigidos o si son bipartitos o no bipartitos, entre otras categorías.

### 1.1.2. Grafos estáticos

Un grafo estático se caracteriza por tener una estructura fija que no cambia con el tiempo. Una vez creados los nodos junto con sus relaciones o aristas, estos permanecen constantes. Este tipo de grafos vienen representados como $G = (V, E)$ donde $V = \{v_1, v_2, ..., v_{|V|}\}$ es el conjunto de nodos y $E \subseteq V \times V$ es el conjunto de aristas.

La mayoría de las veces, un grafo estático representa una instantánea de las relaciones en una red en un momento determinado. Este tipo de grafo es especialmente útil para modelar escenarios en los que las relaciones no cambian con frecuencia o en los que los cambios a lo largo del tiempo no son críticos para el análisis de la red. Un ejemplo de este tipo podría ser una red de trenes, en donde las estaciones son los nodos y las aristas las conexiones entre ellas y donde la adicción de una nueva estación o una nueva vía es muy poco frecuente.

Para caracterizar estos grafos, existen diferentes matrices asociadas a ellos. Entre las más importantes se encuentran por ejemplo la matriz de adyacencia $A \in \mathbb{R}^{|V|\times|V|}$, en la que los valores $A[i][j]$ son igual a $0$ si $(v_i, v_j) \notin E$ y si no son igual a $\mathbb{R}_+$ si $(v_i, v_j) \in E$ para representar el peso de la arista. También existe la matriz de grados $D \in \mathbb{R}^{|V|\times|V|}$, la cual es una matriz diagonal donde $D[i][i] = \sum_{j=1}^{|V|} A[i][j]$ y representa el grado de $v_i$. Y por último está la matriz Laplaciana definida como $L = D - A$, la cual es crucial ya que

encapsula propiedades importantes del grafo a partir de las matrices de adyacencia y grado.

Hay varias alternativas a la hora de representar un grafo estático, por ejemplo, los no dirigidos son aquellos en los que el orden de los nodos en las aristas no es importante, por lo que la matriz de adyacencia es simétrica. En la otra cara se encuentran los grafos dirigidos, en los que el orden de los nodos en las aristas si es importante y existe un nodo origen y un nodo destino para cada una de ellas. Por otro lado, están los grafos bipartitos, en los cuales se pueden dividir los nodos en dos grupos que no estén conectados por ninguna arista, o los grafos con atributos, en los que los nodos, las aristas o ambos tienen características propias.

### 1.1.3. Grafos dinámicos

Al contrario que los grafos estáticos, un grafo dinámico es una estructura que sufre cambios y evoluciona con el tiempo. Se pueden añadir o eliminar nodos y aristas, y los atributos dentro del grafo pueden cambiar dinámicamente. Esta flexibilidad permite a este tipo de grafos modelar relaciones en evolución, capturando cambios en la conectividad o en los atributos de nodos y aristas.

Los grafos dinámicos resultan útiles en situaciones en las que el aspecto temporal es crucial para el análisis. Por ejemplo, en una red social dinámica, los nodos pueden representar individuos y las aristas, amistades. A medida que la gente establece nuevas conexiones o pone fin a las existentes, el grafo dinámico evoluciona para reflejar el cambiante panorama social.

Dentro de los grafos dinámicos se pueden distinguir cuatro tipos:

- Grafos con aristas ponderadas: Estos grafos incorporan información temporal asignando etiquetas a las aristas y/o nodos de una red estática. El ejemplo más básico de este enfoque consiste en etiquetar las aristas de una red estática con las marcas de tiempo que indican cuándo aparecieron.

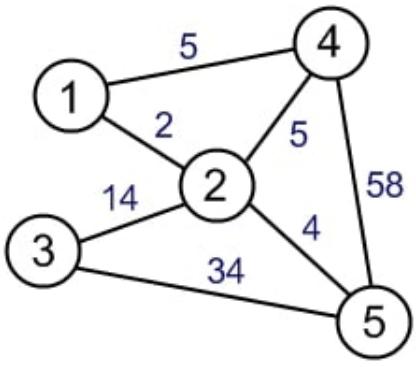



Figura 1.2: Grafo de aristas ponderadas

- Grafos espacio-temporales: Son aquellos en los que los nodos y aristas permanecen estáticos pero sus características pueden cambiar. Un ejemplo concreto de un grafo espacio-temporal podría ser un grafo que represente el tráfico de vehículos en una ciudad. En este grafo, los nodos representarían las intersec-

ciones y las aristas representarían las calles por las que circulan los vehículos. Mientras que los nodos y aristas no varían, las características de este grafo podrían cambiar con el tiempo de varias maneras ocasionado por ejemplo por cambios en la congestión del tráfico, la velocidad promedio o la disponibilidad de carriles.

- Grafos dinámicos de tiempo discreto: Los grafos dinámicos de tiempo discreto, comúnmente conocidos como grafos temporales basados en instantáneas, se caracterizan por ser una sucesión de instantáneas de un grafo estático, con intervalos temporales regulares entre ellas. Estas instantáneas se definen como un conjunto $G^1, G^2, ..., G^T$, donde cada $G^t = V^t, E^t$ representa un grafo estático en el tiempo $t$.

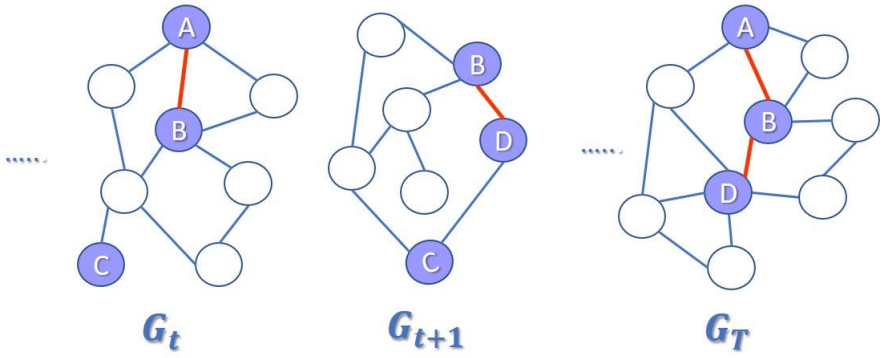



Figura 1.3: Instantáneas de un grafo dinámico discreto

- Grafos dinámicos de tiempo continuo: Diversas metodologías han sido desarrolladas para representar esta categoría de grafos; no obstante, el enfoque más prevalente es el basado en eventos, razón por la cual también son referidos como grafos temporales basados en eventos. Estos modelos se presentan bajo la forma de un par $(G, O)$, donde $G$ representa un grafo estático que encapsula el estado inicial del sistema en el tiempo $t_0$, mientras que $O$ denota un conjunto de observaciones o eventos que comprenden fenómenos como la aparición, modificación o eliminación de nodos o vértices. Existe una vinculación sustancial entre estos grafos y sus predecesores, dado que en cualquier instancia temporal $t \geq t_0$, es posible generar una instantánea $G^t$ mediante la actualización de $G$ con las observaciones $O$ hasta ese momento.

Entre los distintos tipos de grafos dinámicos, el que ostenta la capacidad de almacenar mayor cantidad de información temporal es el de grafos dinámicos de tiempo continuo. Este tipo de grafo dinámico permite la representación y manipulación de una amplia gama de eventos en un tiempo continuo, lo que lo convierte en un recurso invaluable para el análisis de sistemas temporales complejos y dinámicos. Sin embargo, es importante subrayar que su utilización conlleva una mayor complejidad en comparación con los otros tipos de grafos dinámicos.

Es crucial resaltar que, en ciertos escenarios donde la precisión temporal no constituye un requisito imperativo, los otros tipos de grafos dinámicos demuestran también su utilidad. Estas variantes, aunque menos detalladas en cuanto a la representación temporal, ofrecen una estructura más simple y accesible, lo que facilita su implementación y comprensión en determinados contextos de análisis.

## 1.2. Problemas de predicción en grafos

Los desafíos inherentes a la predicción en grafos constituyen un campo de estudio que abarca una diversidad de problemas, cada uno con sus propias complejidades y potenciales aplicaciones. Dentro de este ámbito, emergen tres desafíos principales que requieren una atención especial y una exploración detallada.

El primero de ellos es la clasificación de nodos, una tarea fundamental que implica la asignación de nodos a clases específicas de un conjunto predefinido. Por ejemplo, en una red dinámica, se pueden categorizar los nodos según la evolución del grafo para identificar de manera efectiva valores atípicos o comportamientos inesperados de los nodos, mejorando la seguridad de la red y el monitoreo del rendimiento.

Paralelamente, la predicción de enlaces surge como otro aspecto crítico del análisis de grafos. Este desafío gira en torno a pronosticar nuevos vínculos entre nodos dentro del grafo. Tiene una inmensa importancia en diversos escenarios, como predecir posibles conexiones sociales en plataformas en línea, facilitar sistemas de recomendación de amigos o anticipar interacciones dentro de las redes de transporte. Al anticipar con precisión los enlaces futuros, las empresas y los responsables de la formulación de políticas pueden optimizar la asignación de recursos y la prestación de servicios, mejorando la experiencia del usuario y la eficiencia operativa.

Por último, la complejidad del análisis de grafos se extiende al ámbito de su propia clasificación, lo que presenta la tarea de asignar grafos completos a categorías específicas de un conjunto predefinido. Este problema puede resultar muy útil en aplicaciones como el análisis de redes biológicas para identificar diferentes tipos de estructuras o en la clasificación de documentos en función de la estructura de sus relaciones.

## 1.3. Redes neuronales sobre grafos

Para abordar estos tres problemas propuestos en la sección anterior se han desarrollado diversas técnicas y modelos específicamente diseñados para trabajar con datos en forma de grafos.

En la últimas décadas, el aprendizaje profundo ha tenido un notable éxito en diversos campos debido en gran medida a dos factores clave. El primero de ellos es el rápido avance de los recursos computacionales, en particular la utilización de potentes GPUs y TPUs. Este progreso ha mejorado significativamente la eficiencia y la escalabilidad de estos algoritmos. En segundo lugar, la disponibilidad de grandes cantidades de datos de entrenamiento ha proporcionado a los modelos la información necesaria para aprender patrones complejos y para tener la capacidad de extraer características latentes de datos euclidianos como imágenes, texto y vídeos.

A medida que sus aplicaciones continúan expandiéndose, existe una creciente necesidad de abordar conjuntos de datos que no se ajustan a la estructura euclidiana tradicional tratada por los algoritmos de aprendizaje profundo. Como se ha demostrado en las secciones anteriores, muchos conjuntos de datos del mundo real se representan mejor como grafos, donde las entidades están conectadas mediante relaciones. Los datos en forma de grafo plantean desafíos únicos debido a su naturaleza irregular y desordenada, a diferencia de las imágenes, donde los píxeles están dispuestos

en una cuadrícula o de las frases donde las palabras vienen secuenciadas. Además, los grafos pueden variar en tamaño y tener nodos con, por ejemplo, distintos grados de conectividad, lo que hace que las operaciones estándar de aprendizaje profundo, como las convoluciones, sean menos sencillas de aplicar en este dominio.

En respuesta a estos desafíos y líneas de investigación, ha surgido un gran interés en adaptar estas técnicas de aprendizaje profundo para manejar los datos estructurados en grafos. Inspirándose en modelos exitosos como las redes neuronales convolucionales (CNN) o las redes neuronales recurrentes (RNN), los investigadores han estado desarrollando activamente nuevas metodologías y definiciones de operaciones fundamentales adaptadas a este tipo de datos para resolver los problemas asociados a la predicción en grafos mencionados en la sección anterior.

Posteriormente en el Capítulo 2 se expondrán algunos de los trabajos más importantes en este campo, tanto orientados a grafos estáticos como dinámicos, pero haciendo más hincapié en los orientados a grafos dinámicos, que servirán más adelante para el análisis y mejora del artículo a estudio en este trabajo de Rossi et. al.

## 1.4. *Embedding* de grafos

El *embedding* de grafos o aprendizaje de representación de grafos es una técnica fundamental del aprendizaje profundo en este tipo de estructuras que se utiliza para transformar los nodos y aristas de la red en representaciones vectoriales de baja dimensión. Estas representaciones están diseñadas para capturar aspectos esenciales de la estructura topológica de la red y la información del contenido del nodo o enlace, con el objetivo de codificar las intrincadas relaciones y características de los nodos y aristas en un formato que facilite la resolución de los diversos problemas de predicción en grafos.

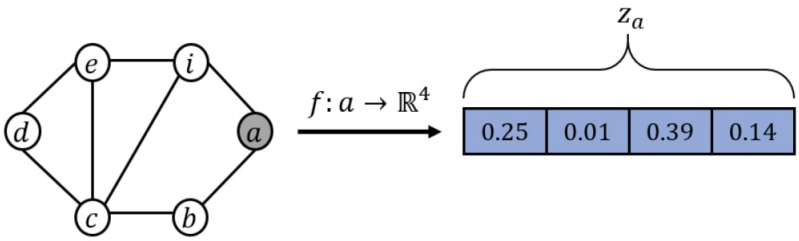



Figura 1.4: Imagen de Khoshraftar and An (2024) sobre el *Embedding* de grafos

Como ya se ha comentado, la ventaja clave de estos métodos es su capacidad de sintetizar datos complejos en forma de red en formas vectoriales concisas, es decir, de baja dimensión. Esta transformación permite aplicar posteriormente algoritmos comunes de aprendizaje profundo ya disponibles para abordar diversos desafíos sobre grafos con diferentes estructuras. Ya sea que la tarea implique predecir atributos de nodos o descubrir estructuras comunitarias dentro de las redes, el embebido de grafos sirve como un paso de pre-procesamiento que mejora la accesibilidad y la utilidad de los datos para aplicaciones posteriores.

El *embedding* de grafos, tanto estáticos como dinámicos, puede ser conceptualizado a través de un marco codificador-decodificador como presentan Kazemi et al. (2020).

En la etapa de codificación es donde ocurre el *embedding* del grafo, es decir, donde los nodos y/o aristas del grafo de entrada se convierten en una representación vectorial de dimensión fija y normalmente baja. Este proceso de codificación es esencial para capturar la información estructural y relacional contenida en el grafo, y para lograrlo se emplean las técnicas de aprendizaje profundo adaptadas a grafos mencionadas anteriormente como las redes convolucionales de grafos o las redes de atención a grafos.

Después de la etapa de codificación, el decodificador asume la responsabilidad de emplear la representación vectorial adquirida para llevar a cabo diversas predicciones, tales como la clasificación de nodos o la predicción de aristas. Fundamentalmente, la función primordial del decodificador radica en la traducción de la representación aprendida en resultados de relevancia y significado.

En el próximo capítulo Capítulo 2), se presentarán algunos de los trabajos más significativos en este campo que ayudarán a entender mejor el artículo a estudio Rossi et al. (2020), ya que este trabajo presenta un nuevo codificador para grafos temporales o de tiempo continuo.

## 1.5. Entornos de aprendizaje

Por último, para contextualizar los grafos y los métodos y enfoques empleados para la resolución de los problemas de predicción asociados a estos, se distinguen dos modalidades fundamentales de aprendizaje en grafos, el inductivo y el transductivo.

El aprendizaje inductivo se caracteriza por su enfoque en la inferencia de patrones generales a partir de un conjunto de datos de entrenamiento, con la finalidad de aplicar esos patrones para realizar predicciones sobre nuevos conjuntos de datos. En el contexto de los grafos, este proceso implica la captura y comprensión de las características y relaciones presentes en un conjunto dado de entrenamiento, con la intención de generalizar dicho conocimiento para hacer frente a datos que no hayan sido observados previamente. Se pueden conceptualizar dos enfoques distintos para comprender este tipo de aprendizaje, uno desde una perspectiva más amplia, en la cual el conocimiento adquirido a partir de un conjunto de grafos se extiende y aplica a otros grafos que el modelo nunca ha visto previamente en el entrenamiento; y otro enfoque en el que durante el proceso de entrenamiento se ocultan selectivamente nodos y aristas, con el propósito de luego generalizar el aprendizaje a nodos y conexiones que no han sido observados durante la fase de entrenamiento.

Por otro lado, el aprendizaje transductivo se concentra en la predicción de etiquetas o características específicas para un conjunto determinado de nodos o aristas dentro de un grafo en particular. A diferencia del enfoque inductivo, el aprendizaje transductivo no busca extender el conocimiento adquirido más allá del grafo de entrada actual; su objetivo se restringe a realizar predicciones dentro del mismo grafo de referencia.

Ambos enfoques poseen sus respectivas aplicaciones y enfrentan desafíos particulares en el ámbito del aprendizaje automático sobre grafos. La elección entre uno u otro depende en gran medida de la naturaleza intrínseca de los datos y de la índole específica del problema que se esté abordando, así como de consideraciones adicionales relativas a la eficacia y la eficiencia de los algoritmos disponibles.

# Capítulo 2

# Trabajos previos

En este capítulo se presentarán de manera exhaustiva los trabajos que preceden al artículo de Rossi et al. publicado en 2020 (Rossi et al., 2020) los cuales están vinculados con el campo de las redes neuronales aplicadas a grafos. Para proporcionar un contexto adecuado, se comenzará con una introducción detallada de diversos métodos asociados con los grafos estáticos, reconociendo su importancia como precursores en esta área de investigación, ya que fueron los que establecieron las bases teóricas y prácticas sobre las cuales se desarrollaron los posteriores enfoques.

Posteriormente, se dedicará una sección más amplia y profunda a los métodos relacionados con los grafos dinámicos. Se analizarán en detalle las características distintivas de estos métodos, así como sus avances y aplicaciones en comparación con los grafos estáticos. El énfasis en los grafos dinámicos responde a que el artículo a estudio propone un nuevo método que aborda este tipo de grafos y a la creciente importancia y complejidad que estos representan.

## 2.1. Redes neuronales sobre grafos estáticos

A principios de la década de 2000 emergió la formalización de las redes neuronales de grafos, del inglés Graph Neural Networks (GNN), un hito que comenzó a tomar forma con trabajos pioneros como el de Gori et al. (2005). Este trabajo destacó al introducir el innovador concepto de redes neuronales basadas en grafos para el reconocimiento de patrones en este tipo de estructuras. Sin embargo, fue la investigación posterior de Scarselli et al. (2008) la que consolidó y amplió este campo, presentando un marco más completo para las GNN que extendía los modelos anteriores para manejar tipos más generales de grafos, como por ejemplo, acíclicos, cíclicos, dirigidos y no dirigidos.

Estos primeros estudios se pueden categorizar como redes neuronales recurrentes de grafos estáticos, debido a que aprenden la representación de un nodo objetivo propagando la información de sus vecinos de manera iterativa hasta alcanzar un punto fijo estable. Este tipo de redes tienen un significado conceptual importante y han influido en investigaciones posteriores sobre otros tipos de redes como por ejemplo las redes neuronales convolucionales de grafos.

Por otra parte y debido al éxito de las redes neuronales convolucionales en dominios como la visión por computadora, se desarrollaron una gran cantidad de métodos que

redefinen el término de convolución para datos en forma de grafo. Las redes neuronales convolucionales de grafos extienden el concepto de operaciones convolucionales más allá de los datos de la cuadrícula a los datos en forma de red. El concepto fundamental implica calcular la representación de un nodo combinando sus propias características con las características de sus nodos vecinos.

A diferencia de las redes neuronales recurrentes de grafos anteriores, las convolucionales emplean múltiples capas de forma secuencial para capturar y refinar progresivamente las representaciones de los nodos a un alto nivel, permitiendo extraer características cada vez más abstractas de las estructuras gráficas. Este tipo de métodos se divide en dos categorías básicas:

- Enfoques espectrales: Las redes neuronales convolucionales espectrales tienen una sólida base matemática fundamentada en el procesamiento de señales de grafos (Shuman et al., 2013; Sandryhaila and Moura, 2013; Chen et al., 2015). Este tipo de redes asume que los grafos son no dirigidos y las convoluciones de grafos se realizan transformando el grafo en el dominio de Fourier mediante la descomposición en valores propios de la matriz Laplaciana del grafo. Esta transformación permite aplicar las operaciones tradicionales de las redes neuronales convolucionales (CNN) a los datos de los grafos.

  Kipf and Welling (2016) presentan una formulación de las redes neuronales convolucionales en el marco de la teoría espectral de grafos ofreciendo la base matemática esencial y métodos computacionales eficaces para crear filtros convolucionales rápidos y localizados en grafos.

  Debido a la descomposición propia de la matriz Laplaciana, estos métodos se enfrentan a tres limitaciones. El primero de ellos se trata de que cualquier perturbación en el grafo resulta en el cambio de los valores propios, lo que hace necesario volver a calcularlos. En segundo lugar, los filtros aprendidos son dependientes del dominio, lo que hace imposible su uso en grafos con una estructura diferente. Y por último, la descomposición de los valores propios requiere una complejidad computacional de $O(n^3)$.

- Enfoques espaciales: Por otro lado, las redes neuronales espaciales operan directamente sobre la estructura espacial de los grafos sin necesidad de transformaciones espectrales. En las GNN basadas en el espacio, las operaciones convolucionales se aplican directamente a las características de los nodos y aristas del grafo, teniendo en cuenta los patrones de conectividad local entre los nodos vecinos. A diferencia de las GNN espectrales, las espaciales no dependen de la descomposición de la base propia y pueden aprender directamente representaciones de nodos basándose en sus vecindarios locales, lo que hace que sean computacionalmente más eficientes y adecuadas para tareas en las que la estructura del grafo es dinámica o evoluciona con el tiempo.

  Un ejemplo de este tipo de redes son por ejemplos las redes de atención de grafos o "Graph Attention Networks (GATs)" propuestas por Velickovic et al. (2017). En este reciente trabajo, se introduce una arquitectura basada en la atención para realizar la clasificación de nodos. Como se puede visualizar en la Figura 2.1 el método consiste en calcular las representaciones ocultas de cada nodo del grafo atendiendo a sus vecinos, siguiendo una estrategia de auto-atención. Esta arquitectura de atención exhibe varias propiedades interesantes: su operación

es eficiente, ya que se puede paralelizarse a través de pares nodo-vecino; puede manejar nodos del grafo con diferentes grados especificando pesos arbitrarios a los vecinos; y el modelo es directamente aplicable a problemas de aprendizaje inductivo, incluyendo tareas en las que el modelo tiene que generalizar a grafos completamente desconocidos.

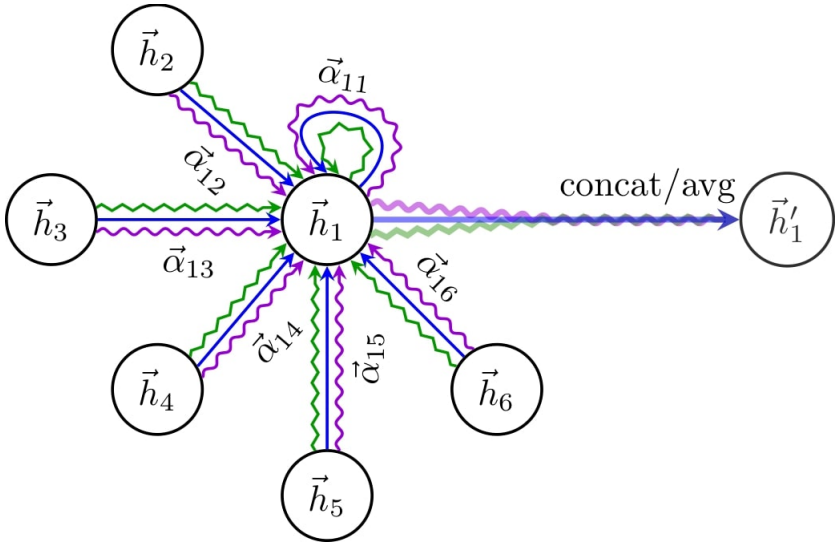



Figura 2.1: Imagen de Velickovic et al. (2017) sobre los GATs

En el contexto del *embedding* de grafos estáticos, los codificadores automáticos destacan en el aprendizaje de representaciones de nodos y aristas aprovechando la información estructural inherente a los grafos, como puede ser la matriz de adyacencia. Como ya se ha comentado anteriormente, este proceso implica codificar los nodos y las aristas en un espacio latente de dimensiones inferiores que encapsula propiedades gráficas esenciales, facilitando tareas como la clasificación de nodos, la predicción de enlaces y la detección de comunidades. Los primeros métodos de este tipo empleaban mayormente perceptrones multicapa para construir tanto el codificador como el decodificador (Cao et al., 2016; Wang et al., 2016) sin embargo, no tardaron en surgir otros que empleaban redes neuronales convoluciones de grafos (Kipf and Welling, 2016) o redes neuronales recurrentes como las de memoria corto-largo plazo (Yu et al., 2018) entre otros tipos de redes.

## 2.2. Redes neuronales en grafos dinámicos

Las redes neuronales sobre grafos dinámicos representan un área de investigación de vanguardia dentro del campo del aprendizaje automático y la inteligencia artificial. A diferencia de los enfoques tradicionales basados en grafos estáticos comentados en la sección anterior, donde los grafos permanecen fijos durante el entrenamiento y la inferencia del modelo, los orientados a grafos dinámicos deben capturan tanto la evolución temporal como la estructural de los nodos y aristas a lo largo del tiempo.

Un comportamiento clave que deben tener las redes neuronales para grafos dinámicos es la capacidad de manejar diferentes estructuras de una manera eficiente. En entornos dinámicos, los nodos y enlaces pueden aparecer, desaparecer o cambiar con el tiempo. Esta naturaleza dinámica requiere que los modelos sean capaces de

aprender de estos cambios y adaptarse a ellos sin un reentrenamiento explícito desde cero. Otra consideración importante es la integración de la información relacionada con el tiempo en los procesos de aprendizaje e inferencia. La dinámica temporal de un grafo puede codificar información valiosa sobre el comportamiento de los nodos, las relaciones y las estructuras comunitarias. Al aprovechar la información temporal codificada en el grafo, estos modelos pueden pronosticar cambios potenciales en la conectividad, ayudando en tareas como la detección de anomalías o el pronóstico de tendencias.

Al igual que en los grafos estáticos, ciertas técnicas de redes neuronales diseñadas para grafos dinámicos también utilizan el denominado *embedding* de grafos. Este enfoque, similar a su aplicación en grafos estáticos, se centra en codificar nodos y aristas en representaciones vectoriales de baja dimensión que conservan características estructurales y semánticas críticas del gráfico. En contextos dinámicos, estas incrustaciones deben ajustarse continuamente para reflejar la estructura cambiante del grafo y la información temporal, lo que requiere que los modelos actualicen iterativamente las representaciones a medida que hay nuevos datos disponibles con el tiempo.

En resumen, las redes neuronales de grafos dinámicos deben capturar los patrones estructurales y temporales en este tipo de redes. Para capturar los patrones estructurales, se suele emplear una red neuronal de grafos, mientras que para los patrones temporales se suele utilizar módulos de series temporales como las redes neuronales recurrentes o los mecanismos de atención posicional.

A continuación se comentarán algunos de los métodos más importantes aplicados a los diferentes tipos de grafos dinámicos para tener una visión más general de las posibles soluciones para resolver problemas de predicción en este tipo de redes. Debido a que cada tipo de grafo dinámico tiene una representación diferente, se subdividirán los métodos según la clase a la que estén enfocados.

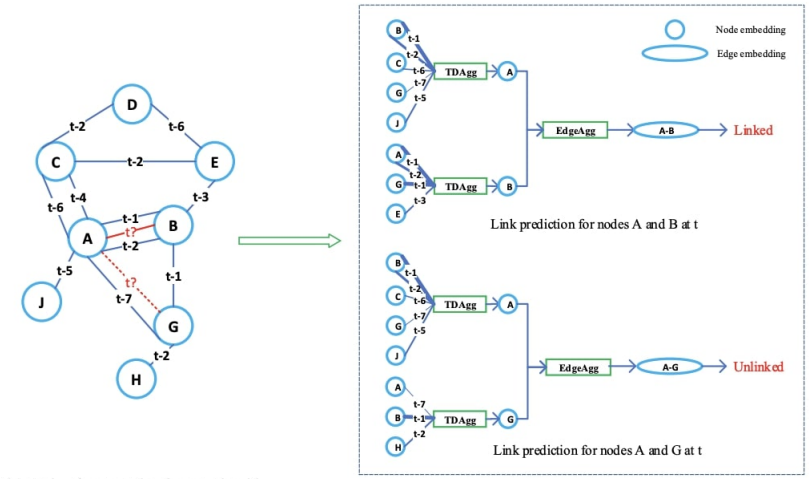
### 2.2.1. Grafos con aristas ponderadas

Los grafos dinámicos, como su nombre sugiere, son estructuras que cambian con el tiempo, lo cual añade un nivel adicional de complejidad a su análisis y comprensión. Uno de los tipos más simples de grafos dinámicos son los grafos con aristas ponderadas, donde cada arista tiene asignado un valor numérico que representa alguna característica o propiedad temporal de la relación entre los nodos que conecta.

Cuando nos encontramos con un grafo dinámico, este se puede simplificar o transformar en una representación más manejable de diferentes maneras. Una forma común de hacerlo es convertirlo en una red estática, donde las conexiones entre nodos se mantienen constantes, aunque ahora con aristas ponderadas que reflejan las relaciones dinámicas originales. Una vez que hemos logrado esta transformación, podemos aplicar diversos métodos de análisis, como las técnicas de redes neuronales especializadas en grafos estáticos anteriormente mencionadas.

Un método que refleja esta solución es por ejemplo Qu et al. (2020), en donde se convierte una red de interacciones en una red de aristas ponderadas utilizando una distribución exponencial. Esto significa que a las aristas que representan interacciones más recientes se les asignan pesos más altos, mientras que las que representan interacciones más antiguas reciben pesos más bajos. Tras esta conversión, se emplea

una red neuronal convolucional de grafos estáticos (Kipf and Welling, 2016) para analizar la red de aristas ponderadas.




(a) A simple example of network with evolving links

(b) TDGNN

Figura 2.2: Imagen de Qu et al. (2020) de su método.

### 2.2.2. Redes neuronales de grafos espacio-temporales

Las redes neuronales de grafos espacio-temporales (STGNN) representan una poderosa herramienta para el análisis de grafos dinámicos, donde tanto los nodos como las aristas experimentan cambios en sus atributos a lo largo del tiempo. Estas redes son especialmente diseñadas para capturar tanto los patrones estructurales como la evolución temporal de los atributos en el grafo.

Una de las principales funciones de las STGNN es la predicción de etiquetas futuras de los nodos. Esto implica entender cómo evolucionarán las características de los nodos a medida que el sistema dinámico evoluciona. Además, estas redes son capaces de estimar etiquetas espacio-temporales en el grafo, lo que significa que pueden prever no solo cómo cambiarán los atributos de los nodos en el futuro, sino también cómo se distribuirán estos cambios a lo largo del espacio y el tiempo en el grafo.

Para abordar estas complejas tareas, las STGNN suelen combinar diferentes arquitecturas de redes neuronales. Por ejemplo, pueden incorporar elementos de redes neuronales recurrentes para capturar la dependencia temporal de los datos, permitiendo que el modelo tenga en cuenta la secuencia de eventos en el tiempo. Al mismo tiempo, pueden emplear capas de redes neuronales convolucionales para detectar patrones estructurales en el grafo, lo que les permite comprender mejor la topología y la disposición de los nodos y las aristas.

La mayoría de estas metodologías capturan eficazmente las dependencias espacio-temporales aplicando convoluciones de grafos para filtrar las entradas y los estados ocultos antes de pasarlos a una unidad recurrente. Seo et al. (2018) presenta dos modelos de este tipo, uno en el que se emplea una red convolucional de grafos propuesta por Defferrard et al. (2016) y una red neuronal recurrente LSTM introducida por Gers et al. (2002); y otro donde se ajusta el método propuesto por Karpathy and

Fei-Fei (2015) para generalizarlo a los grafos en vez de solo a estructuras organizadas como por ejemplo las imágenes.

Estas arquitecturas que hacen uso de redes neuronales recurrentes se enfrentan a importantes retos debido a su propagación iterativa y a su susceptibilidad a la explosión o a problemas de desvanecimiento del gradiente con el tiempo. Estos problemas suelen provocar un entrenamiento lento y una dinámica de aprendizaje inestable. En respuesta, los enfoques basados únicamente en redes neuronales convolucionales presentan un paradigma alternativo para procesar este tipo de datos de forma no recursiva y eficiente, intercalando capas convolucionales de una dimensión con capas convolucionales de grafos para aprender dependencias tanto temporales como espaciales.

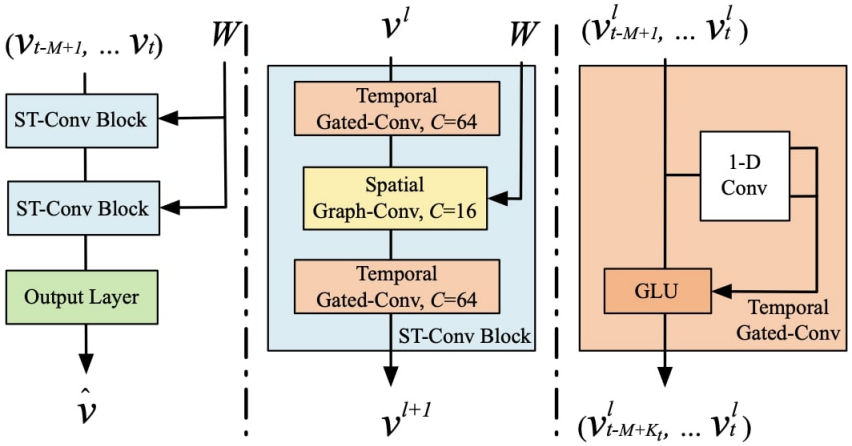



Figura 2.3: Imagen de Yu et al. (2017) de la arquitectura STGCN

Por ejemplo, el método "Spatio-Temporal Graph Convolutional Networks (STGCN)" (Yu et al., 2017) integra capas convolucionales 1-D con redes neuronales convolucionales de grafos (Defferrard et al., 2016; Kipf and Welling, 2016). Como se puede observar en la Figura 2.3, este método forma un bloque espacio-temporal que dispone de capa convolucional de grafos entre dos capas convolucional 1-D cerradas en forma de secuencia.

### 2.2.3. Redes neuronales de grafos dinámicos de tiempo discreto

Como ya se ha comentado en la sección introductoria sobre los grafos, los dinámicos de tiempo discreto están formados por un conjunto de instantáneas que representan el grafo para un tiempo $t$. Uno de los métodos más comunes para abordar problemas con este tipo de grafos es la agregación de las observaciones temporales. Esta aproximación implica convertir el grafo dinámico de tiempo discreto en uno estático agregando las observaciones temporales o las matrices de adyacencia a lo largo de las instantáneas. Posteriormente, se puede aplicar un codificador estático para generar una función de incrustación que capture las características del grafo en su totalidad. Entre las investigaciones que recurren a este enfoque se incluyen, por ejemplo, el trabajo de Liben-Nowell and Kleinberg (2003), en el cual se prescinde de la consideración

temporal y se suman las entradas de las matrices de adyacencia correspondientes a distintas instantáneas, así como el estudio de Sharan and Neville (2008), donde se asigna una mayor ponderación o relevancia a las instantáneas más recientes.

Otros enfoques interesantes son aquellos que emplean la agregación de características estáticas como el propuesto por Yao et al. (2016). Este tipo de métodos, en lugar de generar un grafo estático agregado a partir de las observaciones temporales, aplican un codificador estático a cada instantánea individualmente y luego combinan los resultados a lo largo del tiempo de diferentes formas. Esto permite capturar tanto la estructura global del grafo como las variaciones temporales de los nodos y las relaciones entre ellos.

Por otra parte, métodos como el de Zhu et al. (2016), proponen que el tiempo mismo puede utilizarse como un regularizador en el proceso de aprendizaje. Al imponer una restricción de suavidad a los *embedding* de cada nodo en instantáneas consecutivas, se puede aprovechar la información temporal para mejorar la coherencia y la continuidad en las representaciones de los nodos a lo largo del tiempo. Este enfoque es especialmente útil en situaciones donde se espera que los nodos mantengan propiedades consistentes a lo largo de la evolución del grafo.

Aunque existen algunos otros tipos de métodos, el último a exponer en esta sección es el de los codificadores de modelos secuenciales. Estos modelos se diseñan específicamente para manejar secuencias de instantáneas de grafos y capturar las relaciones temporales entre ellas. Al incorporar información temporal de manera explícita en el proceso de codificación, los codificadores de modelos secuenciales pueden generar representaciones más ricas y contextualmente informadas de los grafos dinámicos. La idea general es generar mediante codificadores los *embedding* $z_v^1, z_v^2, ..., z_v^T$ para cada nodo $v$ de todas las instantáneas del grafo, y una vez calculados emplear una red neuronal recurrente para obtener los *embedding* finales $z_v$. El enfoque descrito anteriormente ha sido propuesto y empleado en diversos estudios, como por ejemplo en el primer modelo del trabajo Seo et al. (2018).

En resumen, los métodos para abordar grafos dinámicos de tiempo discreto son variados y cada uno presenta ventajas y desafíos únicos. La elección del enfoque adecuado depende en gran medida de la naturaleza de los datos y los objetivos del análisis. Sin embargo, todos estos métodos tienen como objetivo común capturar la complejidad temporal de los grafos dinámicos y extraer información significativa que pueda ser utilizada en una variedad de aplicaciones, desde la predicción de eventos hasta la detección de anomalías.

### 2.2.4. Redes neuronales de grafos dinámicos de tiempo continuo

En la actualidad, existen tres enfoques para el modelado de grafos dinámicos continuos. El primero de ellos consiste en los métodos basados en redes neuronales recurrentes, en los que los *embedding* de los nodos se gestionan mediante arquitecturas basadas en RNNs. El segundo enfoque utiliza los denominados procesos temporales puntuales (TPP) junto con redes neuronales que parametrizan estos procesos. Por último, el tercer enfoque utiliza técnicas de *embedding* temporal, en las que las incrustaciones posicionales del tiempo se utilizan para representar el tiempo como un vector.

#### 2.2.4.1. Métodos basados en redes neuronales recurrentes

Los modelos basados en redes neuronales recurrentes utilizan RNNs para actualizar continuamente los *embedding* de los nodos. Una característica común de estos modelos es que cada vez que se produce un cambio en la red, las representaciones de los nodos que interactúan con este evento se actualizan en consecuencia, garantizando que estas permanezcan actualizadas. Hay dos modelos que destacan en esta categoría, el método "Streaming Graph Neural Networks (SGNN)" (Ma et al., 2020), diseñado para codificar redes dirigidas de evolución estricta; y "JODIE" (Kumar et al., 2019), especializado en codificar redes de interacción.

En el método SGNN (Ma et al., 2020) se mantienen diferentes vectores para cada nodo, uno sobre el estado oculto del rol origen, otro sobre el rol destino y otro basado en los dos anteriores. Su arquitectura consiste en un componente de actualización que modifica el estados los nodos involucrados en el evento que ha ocurrido y otro de propagación que transmite esta actualización a los vecinos de los nodos implicados. Estos dos componentes están compuestos a su vez por tres unidades: una unidad de interacción, una de actualización / propagación y otra de fusión.

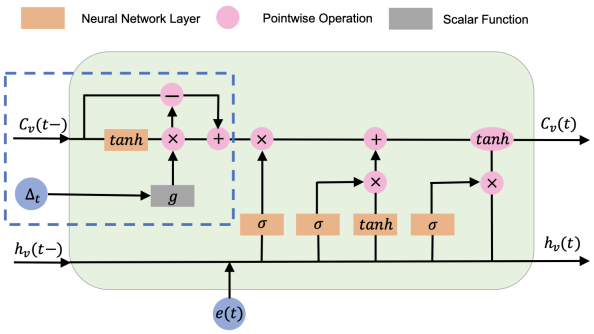



Figura 2.4: Unidad de actualización de Ma et al. (2020).

La unidad de interacción genera una codificación basada en los nodos involucrados en el evento y la de fusión actualiza los estados ocultos de los nodos basándose en los cambios hechos por la unidad de actualización / propagación. La unidad central de actualización (Figura 2.4) genera un nuevo estado oculto para los nodos involucrados mediante una red neuronal recurrente LSTM (Baytas et al., 2017) modificada para funcionar con series de tiempo irregulares. Por otro lado, la unidad de propagación actualiza los estados ocultos de los nodos vecinos utilizando tres componentes: una función de atención, una función de decaimiento temporal y un filtro basado en el tiempo. La función de atención estima la importancia de las conexiones entre nodos, mientras que la función de decaimiento temporal ajusta la magnitud de la actualización en función del tiempo transcurrido desde la última actualización. El filtro basado en el tiempo actúa como una función binaria que filtra las actualizaciones si el nodo receptor tiene información obsoleta, lo que ayuda a reducir el ruido y la eficiencia computacional.

El segundo método mencionado anteriormente (Kumar et al., 2019) consiste en una arquitectura basada en redes neuronales recurrentes para actualizar los *embedding* de cada nodo. En el modelo se utilizan dos RNNs separadas que difieren únicamente en los pesos, una para usuarios y otra para los elementos. JODIE está diseñado es-

pecíficamente para sistemas de recomendación, centrándose en redes de interacción usuario-elemento, sin embargo, puede adaptarse a redes de interacción generales con ligeras modificaciones. Otra característica clave de JODIE es su componente de proyección dentro de la arquitectura, que puede preveer la trayectoria de los *embedding* dinámicos. Este componente predice la posición futura de las incrustaciones de usuarios o elementos y se entrena para mejorar esta capacidad de predicción con el tiempo.

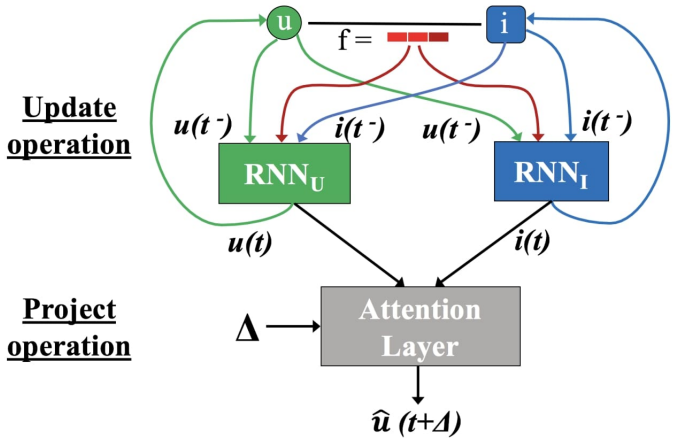



Figura 2.5: Modelo JODIE de Kumar et al. (2019).

#### 2.2.4.2. Métodos basados en puntos temporales puntuales

El método "Know-Evolve"(Trivedi et al., 2017) sirve como base para los posteriores modelos de procesos de puntos temporales de grafos dinámicos. En él se representan los grafos de conocimiento como redes de interacción, lo que se consigue parametrizando un TPP mediante una red neuronal recurrente modificada. Con ligeras modificaciones, el modelo puede adaptarse para su uso con cualquier red de interacción. Sin embargo, como el modelo original está diseñado específicamente para grafos de conocimiento, nos centraremos en su sucesor, "DyREP" presentado por Trivedi et al. (2019).

DyREP utiliza un modelo de procesos puntuales temporales (Figura 2.6) parametrizado por una red recurrente. Estos procesos temporales puntuales capturan tanto la dinámica "de la red", es decir, su evolución estructural; como "en la red", su comunicación entre nodos. Al modelar estas dos dinámicas, se consigue una representación más completa que en la mayoría de los métodos de embebido anteriores.

Los procesos temporales puntuales están modelados por eventos $(u, v, t, k)$ donde u y v son los nodos involucrados en la interacción, t es el tiempo en el que ocurre dicho evento y k indica si es un evento estructural (agregado de arista) o de comunicación entre dos nodos.

#### 2.2.4.3. Métodos basados en el *embedding* del tiempo

Algunos modelos que tratan con grafos dinámicos continuos se basan en métodos de *embedding* temporales, lo que incluye el uso de la codificación posicional para

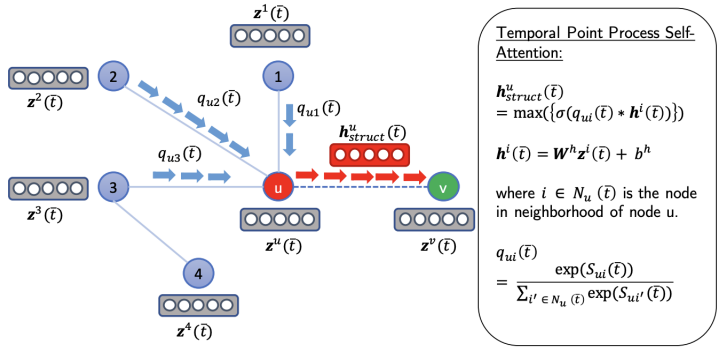



Figura 2.6: Proceso puntual temporal de Trivedi et al. (2019).

representar la dimensión temporal, como se introdujo en Vaswani et al. (2017).

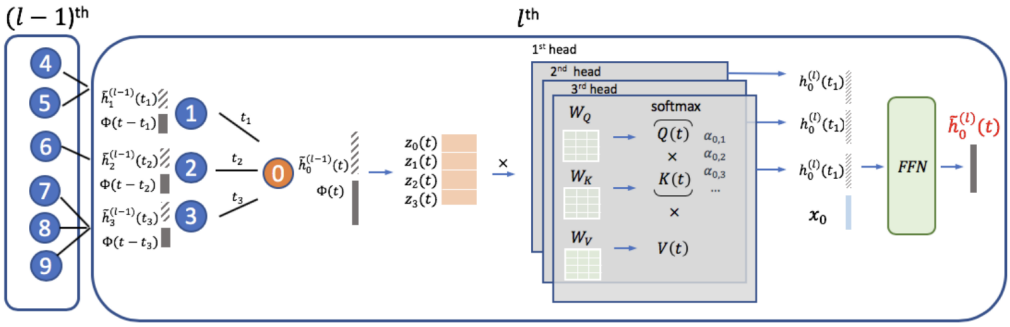



Figura 2.7: Capa TGAT con 3 cabezas de atención de Xu et al. (2020).

El primer método aplicado a grafos dinámicos continuos en usar el embebido de tiempo fue el "Temporal Graph Attention (TGAT)" propuesto por Xu et al. (2020). En él, los autores usan el embebido de tiempo que ellos mismo presentaron en otro artículo (Xu et al., 2019) y que es muy parecido al usado por el método "time2vec" de Kazemi et al. (2019). Básicamente, una capa TGAT concatena las características de los nodos, las de las aristas (opcional) y las temporales de cada nodo vecino y del nodo objetivo y a continuación, aplica una máscara de atención similar a la de GAT (Velickovic et al., 2017).

# Capítulo 3

# Temporal Graph Networks for Deep Learning on Dynamic Graphs

En el presente capítulo, se procederá a realizar un análisis exhaustivo del artículo Rossi et al. (2020), prestando especial atención al modelo y a las mejoras propuestas por los autores. Para iniciar, se proporcionará una visión general del modelo, que incluirá un desglose detallado del flujo operativo. Esto permitirá una comprensión más holística de su funcionamiento, facilitando una apreciación completa de los aspectos técnicos y metodológicos implicados.

A continuación, con el fin de lograr una comprensión integral, se explorarán minuciosamente los cinco módulos que constituyen el modelo: la memoria, la función de mensajes, el agregador de mensajes, el actualizador de la memoria y el módulo de *embedding*. Se detallarán sus características y funciones específicas, proporcionando un análisis profundo de cada componente y su contribución al rendimiento general del modelo.

De igual manera, se discutirán los conjuntos de datos utilizados en la evaluación del modelo, proporcionando una descripción clara del formato y la estructura de los mismos. Además, se explicarán las dos metodologías principales de entrenamiento del modelo: la predicción de enlaces y la clasificación de nodos, destacando las diferencias clave entre ambas y sus aplicaciones específicas. Este análisis permitirá entender cómo cada metodología influye en los resultados y la efectividad del modelo.

Finalmente, se presentarán las sugerencias de mejora planteadas por los autores en el artículo, lo cual permitirá identificar las áreas que requieren mayor atención y posibles caminos que seguir para el desarrollo de este trabajo.

## 3.1. Modelo Temporal Graph Networks

El artículo objeto de estudio en esta investigación corresponde al propuesto por Rossi et al. (2020), donde se introduce el concepto de redes de grafos temporales, conocidas como "Temporal Graph Networks (TGNs)". En dicho trabajo se presenta un marco de trabajo innovador que permite la aplicación de técnicas de aprendizaje profundo a grafos dinámicos de tiempo continuo, los cuales se representan como secuencias de eventos con marcas de tiempo.

De acuerdo con la terminología establecida por Kazemi et al. (2020), un modelo de *embedding* de grafos puede conceptualizarse como un par codificador / decodificador, donde el primero sirve como una función de mapeo de un grafo dinámico a representaciones vectoriales de este, mientras que el segundo utiliza estas representaciones para generar predicciones específicas de la tarea, como la clasificación de nodos o la predicción de aristas. La principal innovación presentada en este estudio es un codificador de redes de grafos temporales (TGN) adaptado a grafos dinámicos de tiempo continuo, representados como una secuencia de eventos. Este codificador produce, para cada momento $t$, los *embedding* de los nodos del grafo denotados como $Z(t) = (z_1(t), \ldots, z_n(t))$.

## 3.2. Flujo operativo del modelo

Antes de examinar minuciosamente los distintos módulos que integran el modelo TGN, es pertinente adentrarnos en una visión holística que abarque el funcionamiento integrado de estos componentes. En este sentido, es crucial comprender cómo interactúan y se complementan entre sí los diversos elementos del modelo, así como su contribución colectiva al logro de los objetivos planteados.

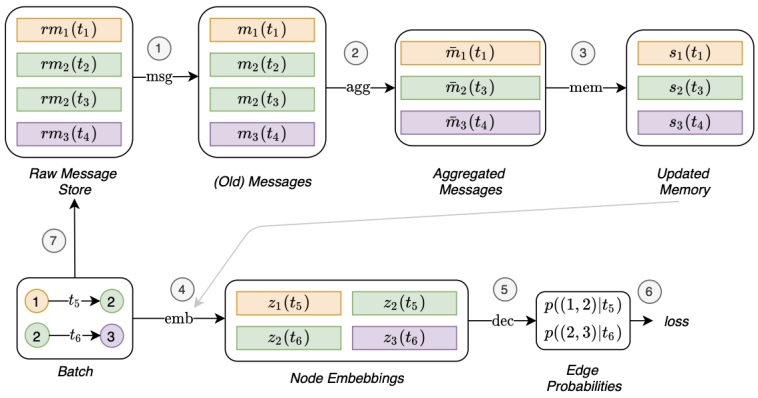



Figura 3.1: Figura 2 de Rossi et al. (2020)

El procedimiento operativo del sistema TGN para el entrenamiento de sus módulos asociados a la memoria involucra una serie de etapas meticulosas y secuenciales. En primer lugar, los datos del *batch* anterior almacenados en el *Raw Message Store* son convertidos en mensajes mediante el módulo de función de mensajes. Posteriormente, en el paso 2, los mensajes pertenecientes a los mismos nodos son agregados mediante el módulo de agregación de mensajes, obteniendo una única representación vectorial de los mensajes de cada nodo del lote anterior.

Una vez calculados y agregados los mensajes, estos son usados por el módulo actualizador de la memoria para obtener los nuevos valores de esta, que posteriormente se usan en el módulo de *embedding* para calcular las representaciones vectoriales de los nodos involucrados en el *batch* actual, no el contenido en el *Raw Message Store*. Tras el cálculo del *embedding* de los nodos, estos se emplean para predecir las probabilidades de las aristas y calcular la pérdida para aplicar el algoritmo de retropropagación al modelo.

Por último, los datos en bruto relativos a las interacciones del lote actual se almacenan en el *Raw Message Store* para su posterior utilización en el siguiente ciclo de entrenamiento. Este meticuloso flujo de operaciones garantiza una gestión eficiente de los datos y recursos del sistema TGN durante el proceso de entrenamiento. Adicionalmente, asegura que el cómputo de los módulos relacionados con la memoria impacte directamente en la función de pérdida, facilitando así la retropropagación de gradientes.

A continuación, se realizará una exposición detallada de los módulos principales que componen el modelo, así como las diversas implementaciones desarrolladas por los autores. A su vez se llevará a cabo una comparación de estas distintas opciones de implementación con base en los resultados obtenidos, con el propósito de evaluar su eficacia y eficiencia relativas.

## 3.3. Módulos principales

Como se ha podido observar en el flujo operativo, el modelo TGN se compone de cinco módulos principales, los cuales desempeñan funciones específicas dentro del sistema. Estos módulos incluyen una unidad de memoria, un componente encargado de la creación de mensajes, otro destinado a la agregación de dichos mensajes, un mecanismo para la actualización de la memoria y, por último, un módulo de *embedding*.

### 3.3.1. Memoria

La memoria del modelo en un momento $t$ consiste en un vector $s_i(t)$ para cada nodo $i$ que el modelo ha visto hasta ese momento. Cuando aparece un nuevo nodo, esta se inicializa a 0 y se actualizará después de cada evento en el que este nodo esté involucrado. Su objetivo es preservar la información histórica del nodo de una forma comprimida para capturar dependencias a largo tiempo para cada nodo en el grafo incluso cuando el modelo ya ha sido entrenado.

En el propio artículo se evaluaron dos modelos idénticos, con la única diferencia de que uno utilizaba memoria y el otro no, con el objetivo de comprobar la eficacia de esta característica. Se comprobó que el modelo que hace uso de la memoria tiene una precisión un 4 % superior que el que no la usa, dando a entender que esta es necesaria para almacenar información a largo plazo sobre los nodos. Además, en estas pruebas se verificó que el uso del muestreo de vecinos más recientes junto a la memoria reducía el número de vecinos necesarios y el número de capas de atención del módulo de *embedding* para lograr mejor rendimiento.

### 3.3.2. Funciones de mensajes

Otro módulo principal es el encargado de formar los mensajes a partir de los eventos que han ocurrido y están almacenados en el *Raw Message Store*. El modelo TGN es capaz de trabajar con tres tipos de eventos y para cada uno de ellos se sigue un proceso diferente:

- Eventos de nodos: Son aquellos en los que está involucrado un solo nodo y están representados como $v_i(t)$, donde $i$ es el índice del nodo involucrado y $v$ el vector de atributos asociado al propio evento. En el caso de que el índice del nodo sea

nuevo para el modelo, se crea un nodo $i$ con las características dadas, si no, se actualizan las características del nodo calculando el siguiente mensaje:

$$m_i(t) = \text{msg}_n(s_i(t^-), t, v_i(t)) \tag{3.1}$$

- Eventos de interacción: Son la representación de una arista temporal $e_{ij}$ entre los nodos $i$ y $j$. En este caso, se calculan dos mensajes, uno para el nodo origen y otro para el destino:

$$m_i(t) = \text{msg}_s(s_i(t^-), s_j(t^-), \Delta t, e_{ij}(t)) \quad m_j(t) = \text{msg}_d(s_j(t^-), s_i(t^-), \Delta t, e_{ij}(t)) \tag{3.2}$$

- Eventos de eliminación: El modelo TGN también soporta eventos de eliminación de nodos y aristas. En el caso de los nodos, simplemente se elimina el nodo escogido y sus aristas salientes y entrantes para que posteriormente no esté involucrado en el *embedding* de otros nodos. En el caso de las aristas, para la eliminación de cada una de ellas se calculan dos mensajes, uno para el nodo origen y otro para el destino:

$$m_i(t) = \text{msg}_{s'}(s_i(t^-), s_j(t^-), \Delta t, e_{ij}(t)) \quad m_j(t) = \text{msg}_{d'}(s_j(t^-), s_i(t^-), \Delta t, e_{ij}(t)) \tag{3.3}$$

En las fórmulas de mensajes anteriores, $s_i(t^-)$ y $s_j(t^-)$ representan la memoria del nodo $i$ y $j$ en el momento anterior al evento y $\text{msg}_s$, $\text{msg}_d$, $\text{msg}_n$, $\text{msg}_{s'}$ y $\text{msg}_{d'}$ son las funciones de mensajes.

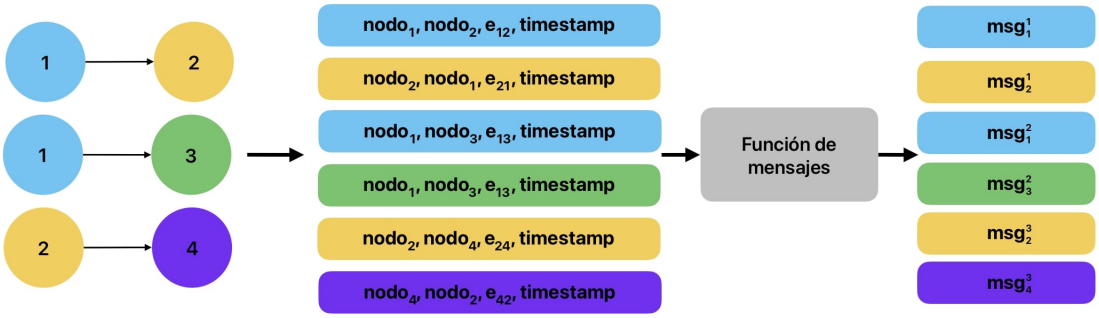



Figura 3.2: Función de mensajes que convierte la información de entrada en mensajes para cada nodo.

Aunque el modelo TGN propuesto es capaz de modelar tres tipos diferentes de eventos, los conjuntos de datos empleados para su experimentación son únicamente de interacciones. Como se ha comentado anteriormente, la información en crudo de estos enlaces se va guardando por lotes en la *Raw Message Store* para posteriormente ser transformada en mensajes por la función correspondiente (Figura 3.2). En el modelo propuesto por los autores del artículo se emplea simplemente una función identidad para cada una de ellas que concatena toda la información de entrada.

### 3.3.3. Agregador de mensajes

Para mejorar la eficacia del modelo, se tuvo en cuenta que pueden existir múltiples eventos en el mismo *batch* que involucran al mismo nodo. Por ello, se creó un me-

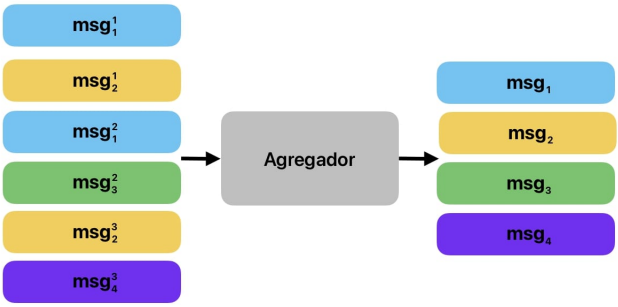



Figura 3.3: Agregador de mensajes de un mismo *batch*.

canismo para agregar los mensajes $m_i(t_1), \ldots, m_i(t_b)$ para $t_1, \ldots, t_b \leq t$ del mismo nodo mediante la función:

$$\bar{m}_i(t) = \text{agg}(m_i(t_1), \ldots, m_i(t_b)) \tag{3.4}$$

La función de agregación, representada por agg, desempeña un papel crucial en este módulo. Para simplificar el proceso, los autores del artículo exploraron únicamente dos soluciones: la utilización del mensaje más reciente denominado "*last*" y el cálculo de la media de todos los mensajes asociados a un nodo específico nombrado como "*mean*". Los resultados revelaron que, si bien la segunda solución ofrecía una precisión ligeramente superior, la opción de emplear el mensaje más reciente conducía a una ejecución del modelo aproximadamente tres veces más rápida.

### 3.3.4. Actualizador de la memoria

Como se ha mencionado anteriormente, la memoria de cada nodo se actualiza cuando ocurre al menos un evento que le involucre de la siguiente manera:

$$s_i(t) = \text{mem}(\bar{m}_i(t), s_i(t^-)) \tag{3.5}$$

Donde mem representa una función aprendible encargada de actualizar la memoria, $\bar{m}_i(t)$ son los mensajes agregados del nodo y $s_i(t^-)$ es la memoria de este antes de ser actualizada. En este contexto, los autores del artículo optan por emplear un tipo de red neuronal recurrente, las Unidades Recurrentes Cerradas (Gated Recurrent Unit (Cho et al., 2014)) como función de actualización, donde los valores de la memoria actúan como *hidden states* de la red neuronal.

### 3.3.5. Módulo de *embedding*

El módulo de *embedding* es el encargado de generar representaciones temporales $z_i(t)$ para cada nodo $i$ en un momento $t$. Debido a que la memoria de un nodo solo se actualiza cuando ocurre una interacción en la que está involucrado, puede ocurrir que esta se quede "estancada", por lo que el objetivo principal de este módulo es evitar este problema denominado "*staleness problem*" (Kazemi et al., 2020).

Aunque existen múltiples opciones para implementar la función de *embedding*, los autores emplean la siguiente:

$$z_i(t) = \text{emb}(i,t) = \sum_{j \in \eta_i^k([0,t])} h(s_i(t), s_j(t), e_{ij}, v_i(t), v_j(t)) \tag{3.6}$$

En la anterior fórmula, $h$ es la función aprendible de *embedding*, y los autores del artículo hicieron pruebas con diferentes soluciones para comprobar su eficacia:

- Función identidad: Hace uso únicamente de la memoria del nodo como *embedding*.

$$\text{emb}(i,t) = s_i(t) \tag{3.7}$$

- Proyección del tiempo: Usa el método de *embedding* empleado en JODIE (Kumar et al., 2019):

$$\text{emb}(i,t) = (1 + \Delta t w) \circ s_i(t) \tag{3.8}$$

  Donde $w$ son parámetros aprendibles por una red neuronal, $\Delta t$ es el tiempo desde la última interacción y $s_i(t)$ es la memoria actualizada del nodo correspondiente.

- Mecanismo de atención temporal de grafos: Siguiendo el método "Temporal Graph Attention (TGAT)" del artículo Xu et al. (2020) en esta solución se emplean $L$ capas de atención de grafos para agregar la información de los vecinos temporales a $L$ distancia.

  A diferencia de la formulación original de esta capa propuesta en TGAT, que no utilizaba características temporales de los nodos, el enfoque de este artículo las incorpora en la representación de entrada de cada nodo. En concreto, definen la representación inicial de un nodo como $h^{(0)}(t) = s_j(t) + v_j(t)$, permitiendo que el modelo aproveche tanto la memoria actual $s_j(t)$ como las características temporales de nodo $v_j(t)$.

- Suma temporal de grafos: Es una manera más simple y rápida de agregar la información de los vecinos en los grafos y hace uso de las siguiente fórmulas:

$$h_i^{(l)} = W_2^{(l)}(h_i^{(l-1)}(t) || \tilde{h}^{(l)}(t)) \tag{3.9}$$

$$\tilde{h}^{(l)}(t) = \text{ReLu}(\sum_{j \in \eta_i([0,t])})W_1^{(l)}(h_j^{(l-1)}(t) || e_{ij} || \phi(t - t_j)) \tag{3.10}$$

  Donde $\phi(\cdot)$ es la codificación de tiempo que se emplea también en el método anterior.

Como se puede apreciar en la figura 3.4 del artículo original, se observa que la solución de proyección temporal deteriora el rendimiento del modelo en comparación con aquella que utiliza directamente la memoria por lo que se descarta su uso directamente. Por otro lado, también se constata que la agregación de información proveniente de los vecinos del grafo resulta efectiva, dado que los dos modelos que la incorporan funcionan mejor que aquel que utiliza la identidad como función de agregación. Profundizando en el análisis, se destaca que la solución que emplea el mecanismo de atención de grafos TGAT supera a aquella que utiliza la suma temporal, aunque con una mayor carga computacional.

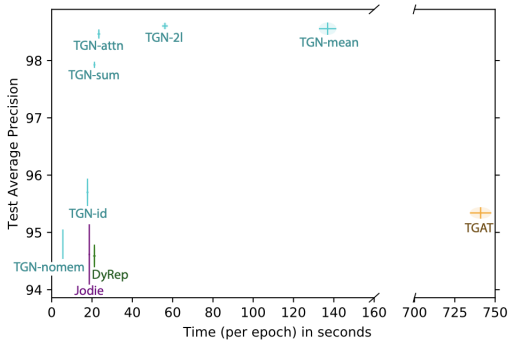



Figura 3.4: Resultados de Rossi et al. (2020)

Además, como ya se ha comentado, los autores corroboraron que, mientras que en el método TGAT original es necesario contar con al menos dos capas de atención para obtener resultados satisfactorios, en el modelo TGN presentado, el uso de memoria permite que únicamente sea necesaria una capa. Esto se debe a que, al acceder a la memoria de los vecinos temporales más cercanos (primera capa), se accede simultáneamente a la información de los vecinos temporales más distantes. Esta estrategia de utilizar una sola capa de atención mejora la eficiencia computacional del modelo.

Por último, es importante señalar que en los dos últimos métodos anteriores que incorporan información de los vecinos, es de gran importancia la implementación del buscador de vecinos temporales. Este buscador será el encargado de escoger los vecinos temporales del nodo en la capa de atención para agregar su información y existen varias metodologías a escoger. En el artículo se optó por emplear el enfoque de los vecinos más recientes en lugar de obtenerlos mediante una distribución uniforme. Esta elección se fundamenta en que mejora su eficacia sin comprometer su rendimiento.

## 3.4. Conjuntos de datos

El estudio original (Rossi et al., 2020) exploró el comportamiento de su modelo TGN en tres conjuntos de datos diversos: Reddit, Wikipedia y Twitter.

En el caso de Reddit, el conjunto de datos presenta un grafo bipartito donde los nodos representan tanto a los usuarios como a los sub-reddits, mientras que las interacciones se manifiestan cuando un usuario comenta en un sub-reddit específico. Por otro lado, Wikipedia exhibe un patrón similar de interacciones, donde los usuarios y las páginas conforman los nodos, y un enlace indica la edición de una página por parte de un usuario. En ambos escenarios, las relaciones están codificadas con un vector de características del texto correspondiente (ya sea comentario o edición), extraídas mediante LIWC (Pennebaker et al., 2001). Además, se etiqueta la presencia de prohibiciones de usuarios en estas plataformas.

En contraste, el conjunto de datos de Twitter ofrece un panorama diferente. Aquí, se presenta un grafo no bipartito donde los nodos representan usuarios individuales y las relaciones se forman a través de retweets. Las características de estas interacciones se capturan mediante una representación vectorial obtenida a través del modelo BERT aplicado al texto del retweet.

Es importante destacar que ninguno de estos conjuntos de datos proporciona características intrínsecas para los nodos en sí mismos. Por lo tanto, en su creación, se asigna el mismo vector cero a cada nodo para mantener la coherencia en el análisis.

## 3.5. Entrenamiento

El artículo presenta un modelo con capacidad para abordar dos tareas específicas de manera eficaz: la predicción de aristas en un contexto de auto-supervisado y la clasificación de nodos mediante un enfoque semi-supervisado.

En el contexto de la predicción de aristas, se emplea un aprendizaje inductivo que garantiza la inclusión de una fracción de nodos como nuevos en los conjuntos de validación y prueba. Esta metodología conlleva una partición exhaustiva de los datos en cinco conjuntos distintos, designados para entrenamiento, validación, prueba, así como otros dos para las interacciones que involucran al menos un nodo recién introducido tanto en las fases de validación como de prueba. A lo largo del proceso de entrenamiento, el modelo sigue un flujo operativo específico que será expuesto detalladamente en el siguiente apartado. Este proceso culmina en la obtención de los *embedding* temporales de los nodos, que posteriormente se utilizan para la estimación de las probabilidades, tanto positiva como negativa, de la existencia de una arista entre dos nodos dados. Con estas probabilidades se calcula la pérdida y se aplica la retro-propagación para el entrenamiento del modelo.

En la tarea relacionada con la clasificación de nodos, se procede a dividir los datos exclusivamente en tres conjuntos distintos: entrenamiento, validación y prueba. Esta decisión se fundamenta en la aplicación de un aprendizaje transductivo, el cual elimina la necesidad de separar nuevos nodos para las fases de validación y prueba. Para llevar a cabo esta tarea, se aprovecha el modelo previamente entrenado en la predicción de enlaces, el cual luego se adapta para la clasificación de nodos. Esto implica un reentrenamiento del modelo mediante el cálculo de los *embedding* temporales de los nodos de origen, que se utilizan para predecir sus respectivas etiquetas mediante un perceptrón multicapa actuando como decodificador. Posteriormente, se evalúa la pérdida comparando los resultados obtenidos del codificador con las etiquetas reales, sobre las cuales se aplica el método de retro-propagación para el ajuste del modelo.

La complejidad de entrenar el modelo TGN del artículo está ligada con los módulos relacionados con la memoria, es decir, el de función de mensajes, el agregador de mensajes y el actualizador de la memoria, ya que no influyen directamente en la función de perdida por lo que no reciben un gradiente. Para solucionar este problema, la memoria debe ser actualizada antes de predecir las interacciones de cada lote. No obstante, actualizar la memoria con una interacción antes de usarla para predecir la propia interacción causa una fuga de información. Para evitar esto, los autores del artículo propusieron actualizar la memoria con los mensajes de los lotes anteriores que están guardados en una memoria denominada "Raw Messages Memory" para luego predecir las interacciones del nuevo lote y por último actualizar esta nueva memoria. Más precisamente, la "Raw Messages Memory" contiene al menos un mensaje en crudo $\text{rm}_i$ para cada nodo $i$ que ha aparecido hasta el momento generado en la última interacción en la que ha estado involucrado dicho nodo.

Otro aspecto importante del entrenamiento es el tamaño de los lotes. Desde la perspectiva de la primera interacción, la memoria estará actualizada, ya que contendrá

información de todas las interacciones anteriores del grafo. Sin embargo, en la última interacción del lote, la memoria no contendrá la información de las interacciones previas de este. Debido a esto, no es recomendable usar tamaños grandes para los lotes y por ello los autores recomiendan usar un tamaño de 200 interacciones.

## 3.6. Mejoras propuestas por los autores

En el propio artículo y como se ha ido comentando en el apartado anterior, los autores dejaron anotaciones con algunas mejoras posibles sobre algunos de los módulos principales del modelo. Estas propuestas implican mejoras que abarcan desde la implementación de una memoria global hasta el uso de otro tipos de funciones aprendibles en por ejemplo el módulo agregador de mensajes. A continuación se listan todas las encontradas en el artículo:

- Aunque el modelo ya cuenta con memorias dedicadas a cada nodo individual, los autores proponen la incorporación de una memoria global de grafos. Esta adición permitiría rastrear la evolución de la red en su conjunto, lo que podría proporcionar una comprensión más holística de la dinámica del sistema.
- En los experimentos realizados, la función de mensaje utilizada fue simplemente la identidad, que consiste en la concatenación de las entradas. No obstante, los autores sugieren explorar funciones de mensaje más elaboradas que involucren una agregación adicional de los mensajes provenientes de los vecinos de los nodos. Esta exploración podría conducir a mejoras significativas en los resultados obtenidos por el modelo.
- A pesar de que en los experimentos se emplearon funciones de agregación que mantenían el mensaje más reciente o calculaban la media de todos los mensajes, se plantea la posibilidad de considerar opciones más diversas. Por ejemplo, se sugiere la exploración de modelos basados en redes neuronales recurrentes (RNNs) o atención a la memoria del nodo. Estas alternativas podrían ofrecer una mayor capacidad de modelado y, en consecuencia, mejorar el rendimiento del modelo en diversas tareas.

Por otra parte, otro aspecto a mejorar del artículo es la parte de experimentación, en donde se propone un uso de otros conjuntos de datos con más variedad de eventos. Además, los tres conjuntos empleados tratan sobre el dominio de las redes sociales, lo que provoca una limitación en términos de diversidad de dominios que puede ser problemático debido a las propiedades intrínsecas de cada tipo de red. Por último, también es posible aumentar el tamaño de los datos, tanto de nodos como de aristas, debido a que en casos reales como las redes sociales existen millones de ellos.

# Capítulo 4

# Desarrollo

En el presente capítulo, se procederá a la introducción de las mejoras propuestas para el modelo Temporal Graph Network (TGN). Estas mejoras abarcan tanto aspectos destinados a incrementar la eficiencia y precisión del modelo, como aquellos enfocados en optimizar su proceso de validación.

Dado que el estudio se enfoca exclusivamente en el modelo temporal TGN, todas las implementaciones adicionales se han realizado sobre el proyecto original disponible en el repositorio de GitHub (https://github.com/twitter-research/tgn). Este enfoque ha sido adoptado con el objetivo de mantener los experimentos bajo las mismas condiciones y parámetros establecidos por los autores del artículo original. Al realizar las modificaciones directamente sobre el proyecto original, se asegura la consistencia y comparabilidad de los resultados obtenidos con aquellos reportados en la investigación inicial.

## 4.1. Propuestas para nuevas funciones de mensajes

Una de las mejoras propuestas por los autores era la de implementar funciones de mensajes diferentes a la de identidad empleada en el artículo para comprobar si esto mejoraba su eficacia.

En este estudio se presenta una nueva función de mensajes diseñada para ser aprendida por perceptrones multicapa. Se realizaron diferentes experimentos para evaluar diversas configuraciones como tamaños de mensajes o número de capas y neuronas, con el objetivo de identificar la estructura más eficaz para optimizar el rendimiento del modelo.

## 4.2. Propuestas para nuevas funciones de agregación

Otro aspecto a mejorar del modelo TGN es el módulo de agregación de mensajes. Los autores probaron dos métodos, uno en el que se obtenía la media de todos los mensajes correspondientes al mismo nodo y otro en el que se mantenía solo el mensaje más reciente para cada uno de ellos.

Sin embargo, estas no son las únicas soluciones al problema de agregación, sino que existen otras que se comprueban en este trabajo:

- Redes neuronales recurrentes: Se implementaron dos tipos de redes neuronales recurrentes, una RNN estándar y una GRU para comprobar su eficacia en la agregación de mensajes.
- Media de los últimos k mensajes: Mezclando las dos soluciones aplicadas por los autores, la idea de este método es obtener los últimos mensajes correspondientes a cada nodo y hacer una media con ellos. De esta manera, podremos obtener información de más mensajes que no sean el último de cada nodo sin comprometer la eficiencia y rapidez del modelo. En cuanto al número de mensajes se probaron con diferentes valores como 10, 20 y 30.

## 4.3. Cambio en el flujo operativo

Aunque el flujo operativo del modelo, tal y como lo delinean los autores, se describe exhaustivamente en la sección 3.2, la aplicación real se desvía de esta metodología prescrita. Un análisis meticuloso del código original del modelo revela una discrepancia entre el flujo propuesto y el implementado. En concreto, la implementación se adhiere más estrechamente al flujo operativo representado en la figura 4.1.

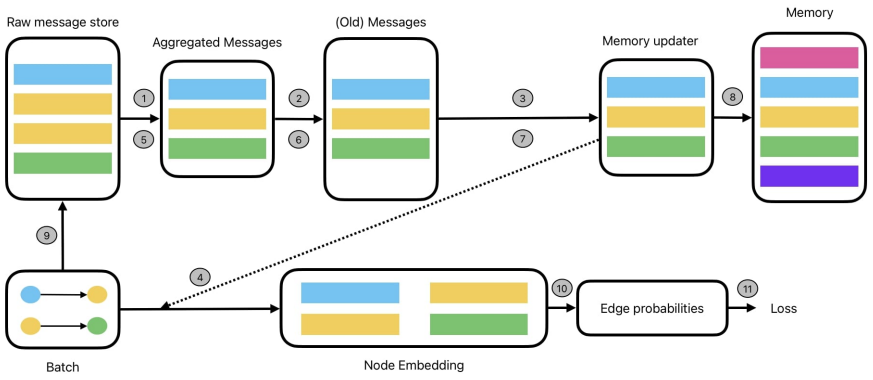



Figura 4.1: Flujo operativo de la implementación original del modelo.

Como se observa en la figura, la tarea de agregación se ejecuta antes que la tarea de función de mensajes. Esta secuencia no afecta al modelo original porque los autores utilizan la función de identidad como función de mensajes. Sin embargo, el orden se vuelve relevante cuando esta función de identidad se sustituye por otra función aprendible, como por ejemplo una red perceptrón multicapa. Como en este trabajo se realizan experimentos con diferentes tipos de funciones, tanto de mensajes como de agregación, se ha corregido este orden para seguir el propuesto en el artículo original.

Además, puede observarse que existe una repetición en la función y agregación de mensajes, así como en el cálculo de los nuevos valores de memoria por parte del actualizador. Esta implementación se debe probablemente a la necesidad de que, durante la pasada inicial, la memoria actualizada debe obtenerse tanto para los nodos positivos como de los negativos, sin necesidad de actualizar realmente los valores de la memoria. En cambio, durante la segunda pasada, sólo se consideran los nodos positivos, ya que son éstos los que se actualizarán en la memoria. Debido a que el tiempo de estos tres módulos es muy pequeño para la configuración original del

modelo, esta repetición de los pasos no supone un gran inconveniente, sin embargo, cuando se emplean funciones de mensajes o de agregación más lentas supone el doble de tiempo que en el caso de realizar estos cálculos una sola vez.

Por estos dos motivos, se modificó el flujo operativo en la implementación del modelo para que siguiera el descrito en el artículo Rossi et al. (2020) de tal manera que se calculase la memoria actualizada para los nodos positivos y negativos pero actualizándose únicamente en memoria los valores de los primeros.

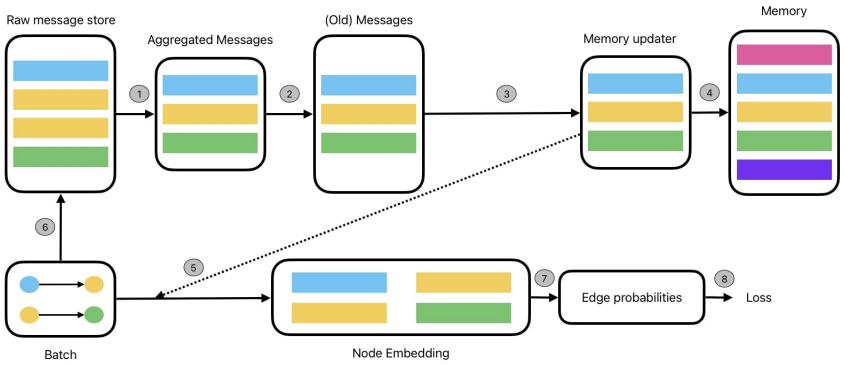



Figura 4.2: Nuevo flujo operativo eliminando la repetición de pasos.

## 4.4. *Embedding* de características de las aristas

Actualmente los mensajes se crean mediante la información de los dos nodos involucrados en la interacción, la marca de tiempo correspondiente y las características del propio enlace. Con respecto a las características de la interacción nos podemos encontrar con dos situaciones, una en la que el número de estas es pequeño, por lo que no se ofrece la suficiente información sobre el enlace; y otra en la que el número es demasiado alto lo que desencadena en un alto coste computacional.

Debido a esto una solución que abarca ambos problemas es la creación de un módulo adicional de embebido de características de las aristas. En los casos en los que el número de características fuera relativamente bajo, este módulo aumentaría la dimensionalidad de estas para otorgar más información al sistema con la finalidad de mejorar su precisión. Por otro lado, en los casos contrarios donde el número de características es alto, disminuiría la dimensionalidad para mejorar el rendimiento del sistema y su coste computacional.

Aunque existen varias opciones de implementación disponibles para este módulo de embebido, se optó por un perceptrón multicapa debido a su simplicidad y eficacia a la hora de facilitar el proceso de integración.

## 4.5. Propuestas para nuevos buscadores de vecinos

Una parte crucial del modelo original en el módulo de embebido es la obtención de los denominados "vecinos temporales". Los vecinos temporales de un nodo son aquellos que han tenido alguna conexión con este en un tiempo anterior al actual $t$ y existen varias metodologías para obtenerlos. En el trabajo original a estudio, se emplean

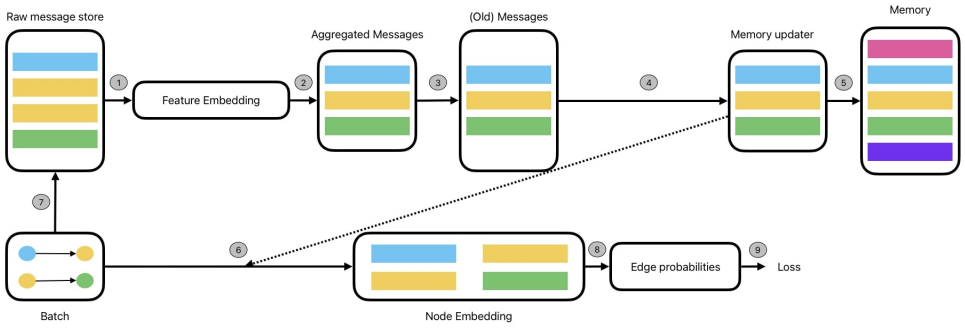



Figura 4.3: Nuevo módulo de embebido de características de las aristas.

dos de los métodos más comunes; el primero consiste en obtener uniformemente $k$ vecinos del nodo que ha tenido hasta el momento actual $t$; por otro lado, el segundo se centra en obtener los $k$ vecinos que han interactuado con el nodo objetivo más recientemente.

Aunque estos dos métodos son los más usados en el área de redes neuronales de grafos temporales, existen otras opciones que se estudiarán en este trabajo para ver si es posible una mejora de la precisión del modelo usando alguna otra alternativa.

La primera opción nueva a explorar es la presentada por Jin et al. (2022), donde se presenta un nuevo método de elección de vecinos temporales denominado NeurTW donde se tienen en cuenta dos motivaciones. La primera de ellas y la más común es la de asignar una mayor probabilidad de muestreo a los vecinos más recientes, ya que tienden a proporcionar más información. Específicamente, dado un nodo en el tiempo $t$, se utilizan las marcas de tiempo de las aristas conectadas a sus vecinos para priorizar aquellos más cercanos al tiempo $t$. Este enfoque emplea un mecanismo de probabilidad basado en el tiempo, modulado por un hiperparámetro $\alpha$, que controla la intensidad del sesgo temporal.

Por otra parte, los autores del artículo proponen que también se debe prestar considerable atención a la conectividad de los vecinos de un nodo ya que estos pueden revelar propiedades diversas y potencialmente significativas dentro del grafo. Para capturar este aspecto, se introduce una probabilidad espacial, que utiliza el grado del nodo como medida de su conectividad. Este sesgo espacial está regulado por otro hiperparámetro, $\beta$, que permite un énfasis controlado de esta parte. Por último, la probabilidad final asignada a cada vecino, considerando estos factores temporales y espaciales, se determina promediando sus probabilidades normalizadas.

Otras opciones de buscador de vecinos son las propuestas en el artículo Wang et al. (2022), donde proponen dos alternativas. La primera de ellas y más sencilla se denomina ”muestreo de vecinos extendido” y en ella se sigue la alternativa de obtener los últimos vecinos pero se propone la adición de saltar $n$ vecinos por cada uno elegido. De esta manera se pretenden obviar las interacciones redundantes y repetidas de un grafo. La segunda de ellas sigue la misma idea que la anterior, salvo que el número de vecinos a saltar es una función aprendible para cada nodo ya que pueden existir momentos del tiempo en los que convenga saltar más o menos vecinos.

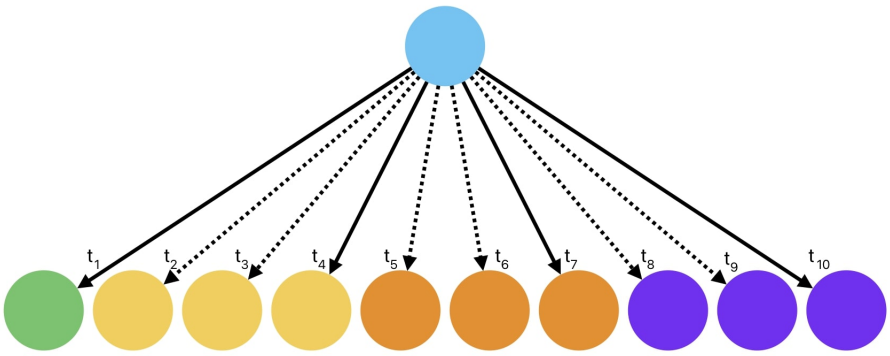



Figura 4.4: Método ”last-skip” con salto de 2 propuesto en Wang et al. (2022).

## 4.6. Propuestas nuevas unidades recurrentes cerradas

Desde la publicación del artículo original que presentaba el modelo TGN, han surgido diversos métodos y tecnologías innovadoras que tienen el potencial de mejorar significativamente el modelo propuesto por Rossi et al. Un ejemplo destacado de estos avances es la evolución de las Unidades Recurrentes Cerradas (GRU), las cuales pueden ofrecer mejoras prometedoras.

Un ejemplo notable de estas nuevas unidades recurrentes es la GRU autorregresiva, propuesta por Jin et al. en 2022 (Jin et al., 2022). En una GRU estándar, los estados ocultos son influenciados tanto por las entradas actuales como por los estados anteriores. Sin embargo, la GRU autorregresiva introduce una modificación clave en este mecanismo, de manera que los estados ocultos $h_t$ dependen únicamente de los estados ocultos anteriores $h_{t-1}$. Esta modificación permite una mejor captura de las dependencias temporales, mejorando la capacidad del modelo para predecir secuencias de datos en función de su propia historia.

Por otro lado, se encuentra la GRU-ODE, introducida por De Brouwer et al. en 2019 (De Brouwer et al., 2019). La GRU-ODE, o "Gated Recurrent Unit with Ordinary Differential Equation", es un modelo innovador diseñado específicamente para manejar datos de series temporales con muestreo irregular, integrando la dinámica de las ecuaciones diferenciales ordinarias (ODE) en la estructura de una red neuronal recurrente GRU. Las redes neuronales recurrentes tradicionales suelen enfrentar dificultades cuando se trabaja con datos que no se muestrean a intervalos regulares, una situación común en muchas aplicaciones del mundo real. La GRU-ODE aborda este desafío modelando la dinámica de los datos en tiempo continuo, lo que permite al modelo adaptarse de manera más efectiva al momento en que ocurren los eventos, mejorando así su precisión y capacidad predictiva.

Aunque existen diversas alternativas adicionales en el campo de las redes neuronales recurrentes, estos dos enfoques —la GRU autorregresiva y la GRU-ODE— son los seleccionados para ser implementados y evaluados en el modelo, con el objetivo de comprobar su eficacia y determinar las mejoras que pueden aportar al rendimiento general del sistema.

## 4.7. Ampliación de los conjuntos de datos

Como se ha mencionado previamente en la sección de Conjuntos de Datos (ver Sección 3.4), los autores del estudio original (Rossi et al., 2020) utilizaron exclusivamente tres conjuntos de datos para el proceso de entrenamiento y validación de su modelo. Esta limitación conlleva a que el modelo solo fue evaluado dentro de los parámetros y características representados por dichos grafos, dando la posibilidad de aplicarlo a nuevos conjuntos más variados para así obtener una mejor validación.

Dada esta problemática sobre la evaluación del modelo y considerando que otros estudios sobre modelos de aprendizaje automático en grafos temporales también enfrentan desafíos similares, se desarrolló un trabajo de referencia significativo denominado "*Temporal Graph Benchmark* (TGB)" (Huang et al., 2024), con el propósito de proporcionar una evaluación más realista, reproducible y sólida de los métodos de aprendizaje automático aplicados a grafos temporales. En este estudio se introdujeron múltiples conjuntos de datos, caracterizados por una amplia variedad en términos de dominios temáticos, número de nodos y aristas, tipologías de los grafos, y hasta variaciones en la medida denominada "tasa de sorpresa", la cual indica el porcentaje de aristas que no se encuentran presentes durante el proceso de entrenamiento. De esta manera, la validación del modelo TGN estará fundamentada en un marco más sólido y amplio, permitiendo una comparación más precisa y significativa con otros enfoques dentro del ámbito de los grafos temporales.

| | **Nodos** | **Aristas** | **Intervalo de tiempo** | **Sorpresa** | **Aristas ponderadas** | **Grafo dirigido** | **Grafo acíclico** |
|---|---|---|---|---|---|---|---|
| Wikipedia | 9227 | 157474 | 30 días | 0.1 | No | Sí | - |
| Twitter | 8161 | 119872 | 7 días | 0.1 | No | Sí | - |
| Reddit | 11000 | 672447 | 30 días | 0.1 | No | Sí | - |

Cuadro 4.1: Conjuntos de datos empleados en el artículo Rossi et al. (2020)

| | **Nodos** | **Aristas** | **Intervalo de tiempo** | **Sorpresa** | **Aristas ponderadas** | **Grafo dirigido** | **Grafo acíclico** |
|---|---|---|---|---|---|---|---|
| **Trade** | 255 | 468245 | 30 años | 0.023 | Si | Si | No |
| **Wikipedia** | 9227 | 157474 | 30 días | 0.108 | No | Si | Si |
| **Review** | 352637 | 4873540 | 21 años | 0.987 | Si | Si | No |
| **Genre** | 1505 | 17858395 | 30 días | 0.005 | Si | Si | No |
| **Coin** | 638486 | 22809486 | 7 meses | 0.120 | Si | Si | No |
| **Reddit** | 11766 | 27174118 | 14 años | 0.013 | Si | Si | No |
| **Comment** | 994790 | 44314507 | 5 años | 0.823 | Si | Si | Si |
| **Flight** | 18143 | 67169570 | 3 años | 0.024 | No | Si | Si |
| **Token** | 61756 | 72936998 | 3 años | 0.014 | Si | Si | Si |

Cuadro 4.2: Conjuntos de datos de TGB (Huang et al., 2024)

Como se puede observar en las tablas 4.1 y 4.2, los conjuntos de datos iniciales presentaban pocas diferencias en términos de tamaño, escala temporal y características como la presencia de aristas ponderadas y la naturaleza de los grafos. Sin embargo, con el desarrollo de TGB y la mejora de los conjuntos de datos empleados para validar los diferentes métodos, se evidencia una diversificación y enriquecimiento considerable. Los nuevos conjuntos de datos abarcan una gama más amplia de dominios, desde el comercio hasta las reseñas, géneros musicales y comentarios en línea, lo que aumenta la representatividad y la capacidad de análisis en diversos campos de estudio. Además, la inclusión de características como las aristas ponderadas y la estructura de los grafos (dirigidos y acíclicos) proporciona una mayor riqueza en los tipos de grafos empleados, lo que mejora la capacidad de analizar los resultados de los modelos.

A continuación se listarán los nuevos conjuntos de datos elegidos del trabajo Huang et al. (2024) para mejorar la validación del modelo TGN original así como de las mejoras introducidas en este trabajo:

- Trade: Esta red internacional de comercio agrícola abarca las naciones de la ONU desde 1986 hasta 2016. Cada nodo representa a una nación miembro de la ONU, mientras que los enlaces indican la suma del valor comercial de todos los productos agrícolas intercambiados entre países. El objetivo es predecir la proporción de los valores del comercio agrícola de una nación hacia otras naciones durante el año siguiente.
- Wikipedia: Este conjunto de datos, idéntico al utilizado en el método original TGN, se representa como un grafo bipartito donde los nodos corresponden a usuarios y páginas, y las conexiones representan las acciones de edición realizadas por los usuarios. El objetivo es predecir en qué página realizará una edición un usuario en un momento dado.
- Review: Esta red bipartita de reseñas de Amazon abarca desde 1997 hasta 2018. Los nodos representan a usuarios y productos, y los enlaces, las calificaciones otorgadas a los artículos, con un peso que representa una puntuación de 1 a 5. La tarea es predecir sobre qué producto realizará una reseña un usuario en un momento dado.
- Genre: Este grafo bipartito representa una red donde los nodos son personas y géneros musicales, y los enlaces indican los géneros que cada usuario ha escuchado, con un peso que representa el porcentaje de pertenencia a ese género. El objetivo es predecir la frecuencia con la que un usuario escuchará cada tipo de género musical durante la próxima semana.
- Coin: Esta base de datos de transacciones de criptomonedas incluye nodos que representan direcciones y enlaces que representan transacciones en un tiempo dado. La tarea principal es predecir los destinos de una dirección en un momento específico.
- Reddit: Similar a los datos usados en el artículo Rossi et al. (2020), esta red bipartita está formada por usuarios y "subreddits". Cada nodo representa un usuario o "subredditz los enlaces corresponden a los comentarios de los usuarios en los "subreddits". El objetivo es aprender la frecuencia de interacción entre estos nodos durante la próxima semana.
- Comment: Este conjunto de datos representa una red de respuestas dirigidas en Reddit, donde los usuarios responden a los hilos de otros usuarios. Cada nodo representa a un usuario y cada enlace dirigido, una respuesta de un usuario a otro. El conjunto de datos abarca desde 2005 hasta 2010 y la tarea principal es predecir si un usuario responderá a otro en un momento específico.
- Flight: En este grafo, los nodos representan aeropuertos y los enlaces, vuelos entre ellos en un día dado. Los nodos están caracterizados por el tipo de aeropuerto, continente, código ISO de la región y su latitud y longitud. Las aristas tienen como característica el número de vuelo. El objetivo es predecir si un vuelo ocurrirá entre dos aeropuertos específicos en un día dado.
- Token: Esta red de transacciones de tokens de criptomonedas incluye nodos que representan usuarios y tokens, y aristas que representan las transacciones

entre ellos, con pesos que indican la cantidad de tokens transferidos. La tarea es predecir la frecuencia con la que un usuario interactuará con varios tipos de tokens durante la siguiente semana.

Como se puede observar en la descripción de cada conjunto de datos, los objetivos de cada grafo se pueden clasificar en dos tipos: la predicción de enlaces y la predicción de afinidad entre nodos. La primera de estas tareas es análoga a la predicción de aristas realizada por el modelo original TGN; no obstante, la segunda no coincide exactamente con la tarea de clasificación de nodos propuesta por Rossi et al. (2020).

### 4.7.1. Preparación de los datos para la predicción de aristas

Como se ha comentado anteriormente, el modelo TGN esta diseñado para llevar a cabo dos tareas, y una de ellas es la predicción de aristas o enlaces. Debido a que el conjunto de datos de Twitter no está disponible de forma gratuita, para los experimentos relacionados con esta tarea que se realicen durante el trabajo, se usaran los de Wikipedia y Reddit del artículo Rossi et al. (2020) y los de Wikipedia, Review, Coin, Comment y Flight mencionados anteriormente del artículo de Huang et al. (2024).

El primer paso es entender el formato de los datos para poder convertirlos a la estructura de entrada que precisa el modelo TGN. Todos los datos vienen en formato .csv donde cada línea representa una interacción entre dos nodos en un tiempo dado y con una estructura diferente para cada conjunto. El modelo TGN necesita como entrada de un "DataFrame" de la librería de "Pandas" con la siguiente información de las interacciones: identificadores del nodo origen y destino, tiempo de la interacción y un identificador de la interacción. Adicionalmente necesita como entrada dos "arrays" de "numpy" que contengan las características de cada aristas y de cada nodo del grafo.

Debido a que los datos del marco de trabajo TGB vienen en un formato diferente, es necesario crear unas funciones específicas en Python para convertirlos al formato requerido por el modelo TGN. Una vez convertida la información necesaria al formato correcto ya podemos aplicar el modelo a estos grafos y así tener más conjuntos de datos sobre los que probar la tarea de predicción de enlaces del modelo.

# Capítulo 5

# Resultados

Este capítulo presenta los resultados obtenidos al aplicar las diversas soluciones propuestas en el capítulo anterior al modelo original. El objetivo es determinar qué modificaciones mejoran la eficiencia o la velocidad del modelo y cuáles, por el contrario, degradan su rendimiento. Mediante la evaluación sistemática del impacto de cada solución, buscamos proporcionar una evaluación exhaustiva de su efectividad y ofrecer recomendaciones sobre qué enfoques deben adoptarse o evitarse.

Todos los resultados presentados a continuación provienen de cinco ejecuciones independientes de cada modelo correspondiente, realizadas en una computadora MacBook Pro equipada con un procesador M2 Pro y memoria de 16 GB. Cabe mencionar que como el modelo TGN presenta una única forma de entrenamiento autosupervisado, la predicción de enlaces, estas pruebas se realizarán en una primera instancia sobre esta tarea, guardando los modelos con mejores resultados para su posterior uso en la segunda tarea de clasificación de nodos. Estos experimentos utilizan un conjunto de datos simplificado del original de Wikipedia, que contiene 81,899 interacciones entre 5,065 nodos.

El uso de este conjunto de datos es necesario debido a las altas demandas computacionales de ejecutar el modelo en conjuntos de datos más grandes, lo que haría que realizar un gran número de pruebas fuera excesivamente prolongado. De esta manera, podemos obtener resultados significativos y comparables en un tiempo razonable, facilitando la identificación de las modificaciones más efectivas para mejorar el modelo original.

Para comprobar el rendimiento de las mejoras en escenarios de aplicación real y a gran escala, se evaluará la eficacia de los mejores modelos nuevos en los conjuntos de datos completos de Wikipedia, Reddit y Review. Esta evaluación se realizará considerando que, en el caso del conjunto de datos de Review, el tiempo de ejecución es de varios días, y para los demás conjuntos de datos, el tiempo de ejecución también se prolongaría durante varios días.

Por último se presentará una sección en la que los mejores modelos entrenados para la predicción de enlaces en los conjuntos de datos de Wikipedia y Reddit se emplearán en el entrenamiento semi-supervisado de clasificación de nodos para comprobar si las mejoras también han supuesto un aumento de la precisión en esta tarea.

## 5.1. Preparación del conjunto de datos simplificado

Antes de comenzar con la experimentación de las nuevas propuestas, se ejecutaron las pruebas del modelo original descritas en el artículo de Rossi et. al. en un ordenador personal MacBook Pro equipado con un procesador M2 Pro sin tarjeta gráfica para comprobar su correcto funcionamiento y el tiempo que consumían cada una de ellas. Como era previsible, a mayor número de nodos e interacciones, mayor tiempo de ejecución, lo que hacía inasumible realizar el número de pruebas requeridas para la comprobación de las nuevas implementaciones en los conjuntos de datos originales del artículo ni en los del *Temporal Graph Benchmark*" (Huang et al., 2024). Con respecto al conjunto de datos de Wikipedia, el tiempo de ejecución del modelo varía entre 20 y 30 minutos, dependiendo del número de épocas ejecutadas. Para el conjunto de datos de Reddit, este tiempo se incrementa a entre 1 hora y media y 2 horas. Finalmente, en el conjunto de datos Review el tiempo de ejecución puede extenderse hasta una semana.

Debido a esto, se propuso realizar una simplificación del conjunto de datos más pequeño disponible, el de Wikipedia. El código proporcionado en el Anexo A simplifica el conjunto de datos de Wikipedia, reduciendo su número de usuarios a la mitad para mejorar la rapidez en las pruebas. Primero, selecciona aleatoriamente el 50 % de los usuarios únicos y filtra el DataFrame graph_df para incluir solo las interacciones de estos usuarios y las páginas asociadas. Luego, reasigna índices a los usuarios ('u') y páginas ('i'), comenzando desde 1, y organiza el DataFrame según la columna de tiempo ('ts'). Finalmente, ajusta el índice ('idx') para que empiece desde 0 y retorna el DataFrame simplificado junto con las características de las aristas (edge_feat) correspondientes a las nuevas entradas filtradas.

Una vez simplificado este conjunto de datos ya se pueden realizar las pruebas necesarias para comprobar el funcionamiento de las nuevas propuestas. Una vez ejecutadas en estos datos simplificados se probarán las que mejor resultados hayan dado con los conjuntos de datos completos propuestos en la Sección 4.7.

## 5.2. Funciones de mensajes y de agregación

Conforme se mencionó en la sección correspondiente del capítulo anterior, se propuso la utilización de un perceptrón multicapa como función de mensajes con el fin de verificar si su implementación podría mejorar la precisión del sistema.

| | Original | MLP 100 | MLP 200 | MLP 500 | MLP 1000 |
|---|---|---|---|---|---|
| Test AP | 0.98292 | 0.97397 | 0.98226 | 0.98246 | 0.98297 |
| New Node Test AP | 0.97697 | 0.97397 | 0.97524 | 0.97732 | 0.97513 |

Cuadro 5.1: Resultados de la precisión de test para el modelo original y los modelos con función de mensajes MLP con diferentes tamaños de salida.

Sin embargo, como se puede apreciar en la Figura 5.1 y en la Tabla 5.1, la integración de este perceptrón multicapa denominado "tgn-message-function-mlp-X" en las gráficas con diferentes dimensiones de salida no representa una mejora en ninguno de los aspectos evaluados del modelo. Por consiguiente, se concluye que su uso resulta ineficiente y no aporta beneficios sustanciales al rendimiento global del sistema.

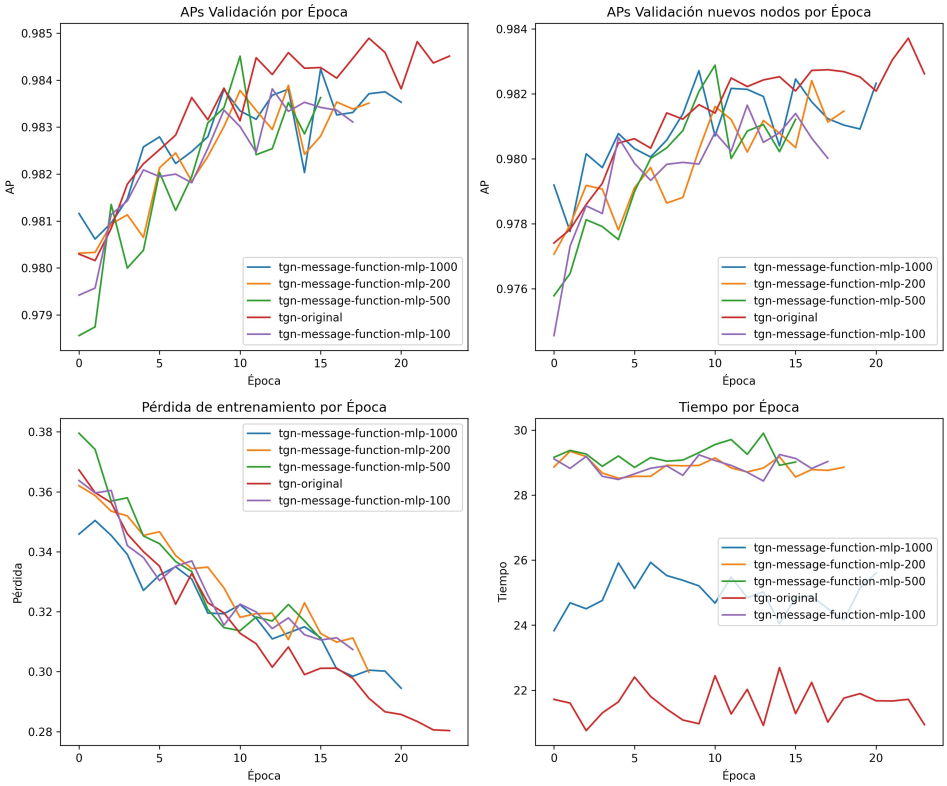



Figura 5.1: Resultados de TGN con función de mensajes MLP.

Por otro lado, en la figuras 5.2 y 5.3 se encuentran los resultados de los modelos con las nuevas funciones de agregación de mensajes. La primera de ellas, denominada "tgn-message-aggregator-latXmean" en la figura, es el resultado de la media entre los mensajes de las últimas $X$ interacciones de un nodo y se emplean diferentes valores de $X$ para observar cual de ellos otorga mejores resultados al modelo. La segunda de ellas, "tgn-message-aggregator-rnn", emplea una red neuronal recurrente por donde pasan secuencialmente los mensajes asociados al mismo nodo para lograr la agregación y la última, "tgn-message-aggregator-gru", utiliza una unidad recurrente cerrada para este proceso.

| | Original | Last10mean | Last20mean | Last50mean |
|---|---|---|---|---|
| Test AP | 0.98292 | 0.98389 | 0.98334 | 0.98498 |
| New Node Test AP | 0.97697 | 0.97722 | 0.97577 | 0.97783 |

Cuadro 5.2: Resultados de la precisión de test para el modelo original y los modelos con función de agregación "lastXmean" con diferentes valores de X.

Como se puede observar en la Figura 5.2 y en la Tabla 5.2, las ejecuciones que emplean el método de agregación "lastXmean" para diferentes valores de $X$ superan en precisión al modelo original TGN que emplea el método de escoger únicamente la última interacción de cada nodo. A medida que se va aumentando el valor de $X$ la precisión del modelo aumenta, por lo que el mejor resultado lo obtiene el que emplea el agregador "last50mean". Con respecto al tiempo de ejecución, se puede observar como a medida que se aumenta el número de mensajes sobre los que hacer la media, el tiempo también lo hace. Por estos dos motivos, el caso ideal es encontrar un punto

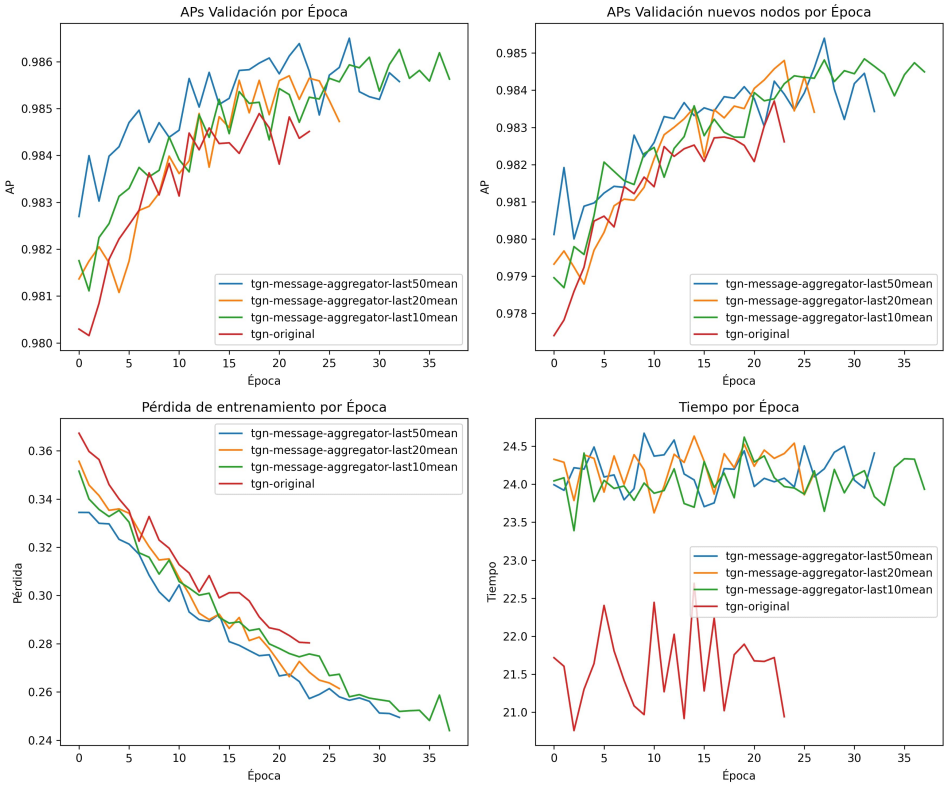



Figura 5.2: Resultados de TGN con agregador "lastXmean".

intermedio que escoja un número representativo suficiente de mensajes pero que no aumente demasiado el tiempo de ejecución.

| | Original | RNN | GRU |
|---|---|---|---|
| Test AP | 0.98292 | 0.98453 | 0.98610 |
| New Node Test AP | 0.97697 | 0.97658 | 0.98046 |

Cuadro 5.3: Resultados de la precisión de test para el modelo original y los modelos con función de agregación RNN y GRU.

Por último, la Figura 5.3 y la Tabla 5.3 muestran como los agregadores basados en una red neuronal recurrente y en una unidad recurrente cerrada también mejoran la precisión del sistema en especial el modelo que emplea la GRU. No obstante, esta mejora supone un gran aumento del tiempo de ejecución debido a la cantidad de parámetros empleados en la retropropagación, lo que hace que estos métodos sean descartados para la agregación de mensajes.

## 5.3. Cambio del flujo operativo

En cuanto al nuevo flujo de trabajo propuesto anteriormente, aunque la base teórica presentada en el artículo original (Rossi et al., 2020) sugiere que podría ser una aplicación valiosa, los resultados indican lo contrario, y en ellos se puede observar por qué los autores originales eligieron un flujo de trabajo diferente en su código.

Para comprobar si este nuevo flujo mejoraba el rendimiento del sistema sin afectar a

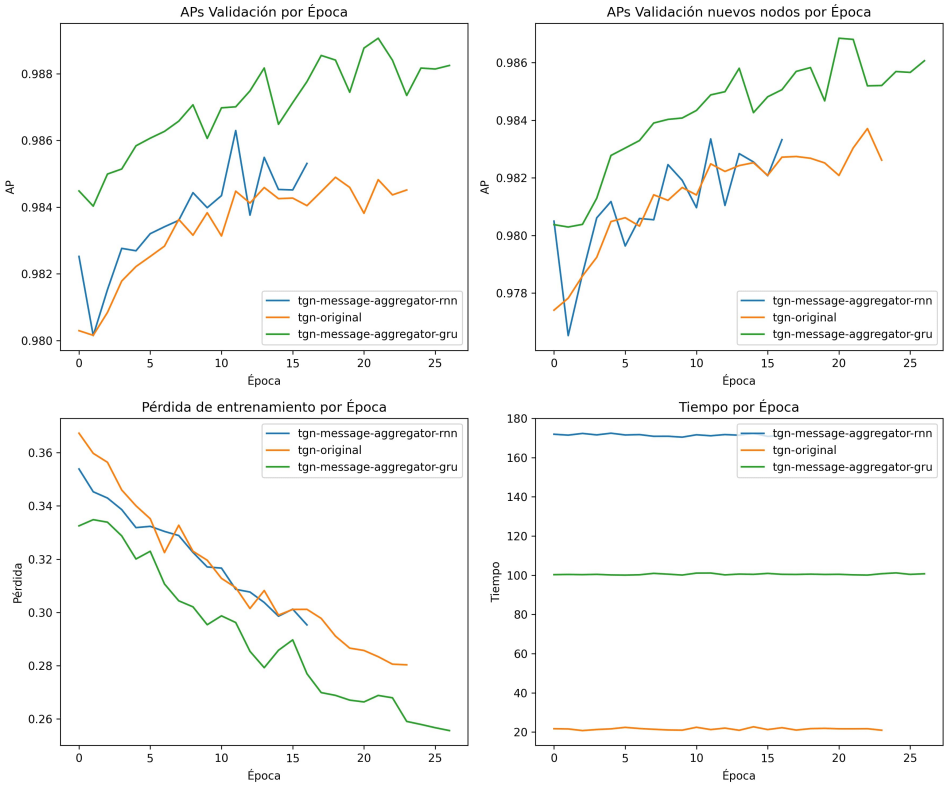



Figura 5.3: Resultados de TGN con agregador RNN y GRU.

su precisión se probó a usarlo con el modelo que emplea una GRU como agregador de mensajes, ya que era la que mejor precisión ofrecía pero peor tiempo de ejecución tenía. Como se ilustra en la figura 5.4, aunque el nuevo flujo de trabajo nombrado como "tgn-message-aggregator-gru-new-workflow" reduce el tiempo de ejecución en casi un 25 %, compromete la precisión del modelo de forma drástica. Esta disminución de la precisión hace que el nuevo flujo de trabajo no sea adecuado como mejora.

## 5.4. *Embedding* de características de las aristas

En cuanto a la nueva funcionalidad de *embedding* de características de aristas antes de la creación de los mensajes, la Figura 5.5 y la Tabla 5.4 ilustran como el perceptrón multicapa implementado y nombrado como "tgn-feature-embedding-X" no mejora sustancialmente el modelo para ningún valor de X mientras que aumenta ligeramente el tiempo de ejecución, lo que lo hace descartable como mejora.

| | Original | F.E. 50 | F.E. 100 | F.E. 200 | F.E. 500 |
|---|---|---|---|---|---|
| Test AP | 0.98292 | 0.98353 | 0.98308 | 0.98386 | 0.98350 |
| New Node Test AP | 0.97697 | 0.97558 | 0.97755 | 0.97839 | 0.977349 |

Cuadro 5.4: Resultados de la precisión de test para el modelo original y los modelos con módulo de embebido de características de diferentes dimensiones.

No obstante, estos resultados se han obtenido de un conjunto de datos reducido y cuyas interacciones están etiquetadas únicamente con 172 características las cuales ya

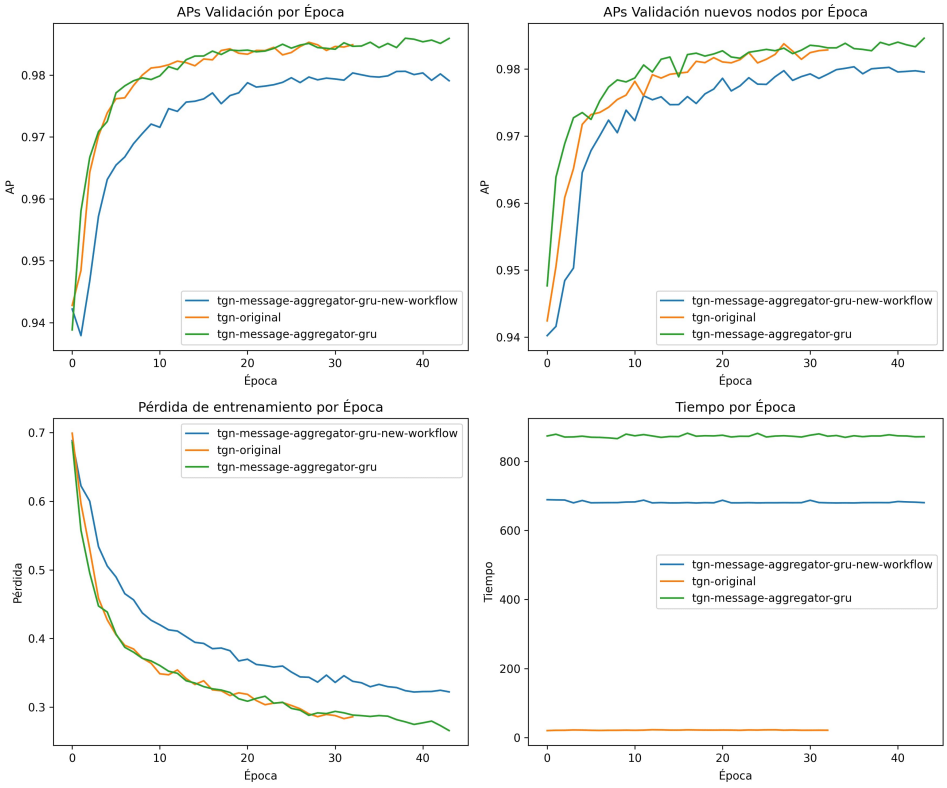



Figura 5.4: Resultados de TGN con agregador GRU y nuevo flujo operativo.

han sido procesadas anteriormente por modelos de redes neuronales, lo que hace que no sean un buen objeto a estudio, y debido a que ningún otro conjunto de datos de los disponibles contiene un mayor número de características no es posible comprobar de manera vigorosa la eficacia del nuevo módulo. Por otra parte, cabe mencionar que la utilidad de este puede ser necesaria para casos en los que las características vengan en un formato menos manejable o tengan un mayor número.

## 5.5. Diferentes tipos de GRU

Otro aspecto a comprobar son los diferentes tipos de unidades recurrente cerradas mencionadas en la Sección 4.6 para ver si actúan mejor como actualizador de la memoria que la unidad GRU original.

| | Original | Auto-GRU | GRU-ODE |
|---|---|---|---|
| Test AP | 0.98292 | 0.98484 | 0.95551 |
| New Node Test AP | 0.97697 | 0.97834 | 0.95195 |

Cuadro 5.5: Resultados de la precisión de test para el modelo original y los modelos con diferentes tipos de GRU como actualizador de memoria.

Como se observa en la Figura 5.6 y en la Tabla 5.5, la GRU-ODE denominada en la gráfica como ”tgn-memory-updater-gru-ode” decrementa la precisión del modelo significativamente, lo que hace que no sea factible para una mejora. Por otra parte, la auto-GRU nombrada como ”tgn-memory-updater-auto-gru” mejora ligeramente am-

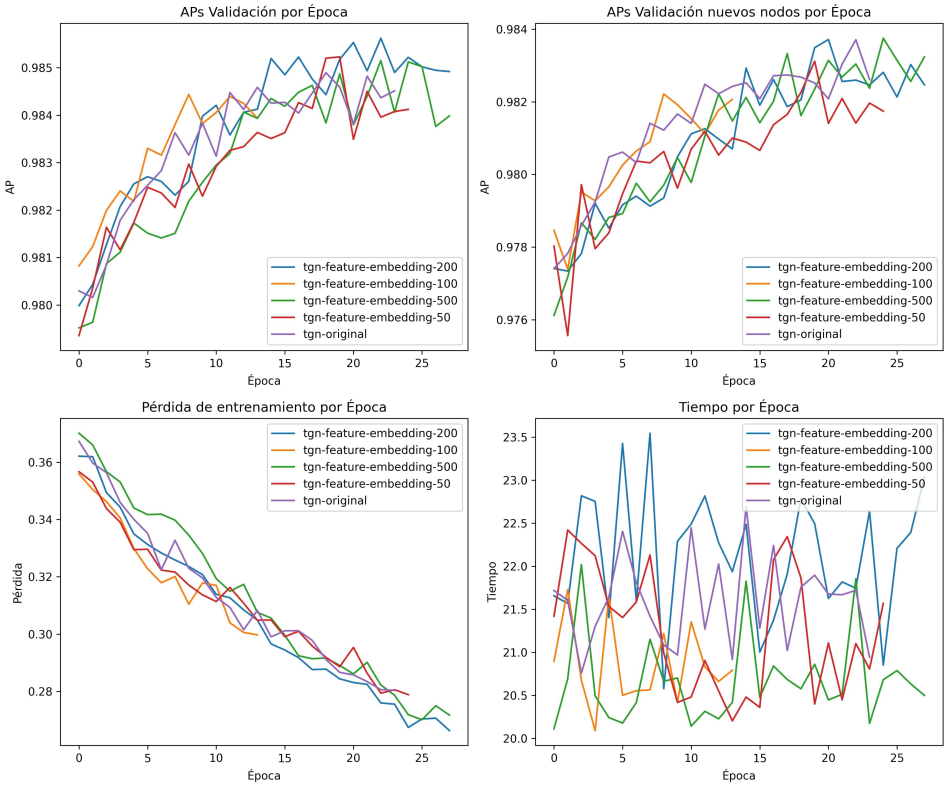



Figura 5.5: Resultados de TGN más *embedding* de características de las aristas.

bas precisiones, tanto la inductiva como la transductiva, y mejora aunque en poca medida el tiempo de ejecución.

## 5.6. Buscadores de vecinos

Por último, con respecto a los nuevos buscadores de vecinos mencionados en el capítulo anterior, se comprobó la eficacia de dos de ellos. Primero se probó el denominado "NeurTW" propuesto por Jin et al. (2022) con diferentes valores de los hiperparámetros que controlan la intensidad de los sesgos, sin embargo, ninguno de ellos mejoró el modelo original TGN. Esta incompatibilidad con el modelo se puede deber a que su uso está enfocado a métodos de caminos aleatorios con el del propio artículo o al conjunto de datos empleado.

Por otra parte, se implementó y probó la primera propuesta del artículo Wang et al. (2022) que propone una modificación del método de escoger los últimos vecinos añadiéndole un salto de $X$ nodos cada vez que se escoge uno (Figura 4.4).

| | Original | Last-skip 1 | Last-skip 2 | Last-skip 3 | Last-skip 4 |
|---|---|---|---|---|---|
| Test AP | 0.98292 | 0.98348 | 0.98315 | 0.98342 | 0.98164 |
| New Node Test AP | 0.97697 | 0.97668 | 0.97719 | 0.97685 | 0.97389 |

Cuadro 5.6: Resultados de la precisión de test para el modelo original y los modelos con el buscador de vecinos "last-skip" con diferentes valores de salto.

Como se puede observar en la Figura 5.7, el método con 2 saltos denominado "tgn-

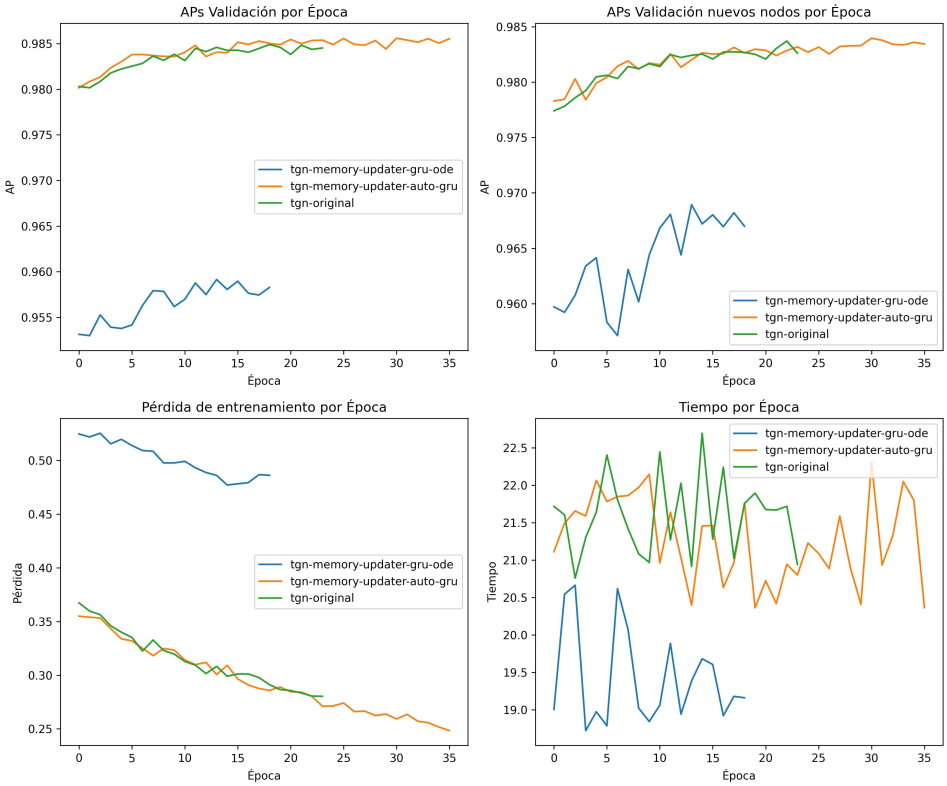



Figura 5.6: Resultados de TGN con diferentes tipos de GRU's.

neighbor-finder-last-skip-2" supera significativamente la precisión del modelo en nodos ya vistos y la mejora también ligeramente para la predicción de nuevos nodos. Por otro lado, el tiempo de ejecución permanece constante o incluso disminuye en algunos casos. Aunque para este conjunto de datos el número de saltos que ha ofrecido mejores resultados ha sido 2, es posible que para otros conjuntos con un mayor número de interacciones, el número óptimo sea más alto.

## 5.7. Resultados de los mejores modelos en conjuntos de datos completos

Tras obtener los resultados de las nuevas implementaciones en el conjunto de datos simplificado de Wikipedia y evaluar su eficacia para mejorar el modelo, estamos preparados para pasar a la siguiente fase de nuestra investigación. Esta fase consiste en probar los modelos más prometedores en un conjunto de datos más amplio. En concreto, utilizaremos estos modelos optimizados en el conjunto de datos completo de Wikipedia, así como en los conjuntos de datos de Reddit y Review. El objetivo de estas pruebas exhaustivas es determinar si la precisión de los modelos ha mejorado realmente como resultado de las nuevas modificaciones e implementaciones.

Durante el proceso de selección del nuevo modelo se dio prioridad a las implementaciones que en los resultados obtenidos en el capítulo anterior obtuvieron mayor precisión que el modelo original TGN sin aumentar significativamente su tiempo de ejecución. Las mejoras específicas incorporadas al nuevo modelo son las siguientes:

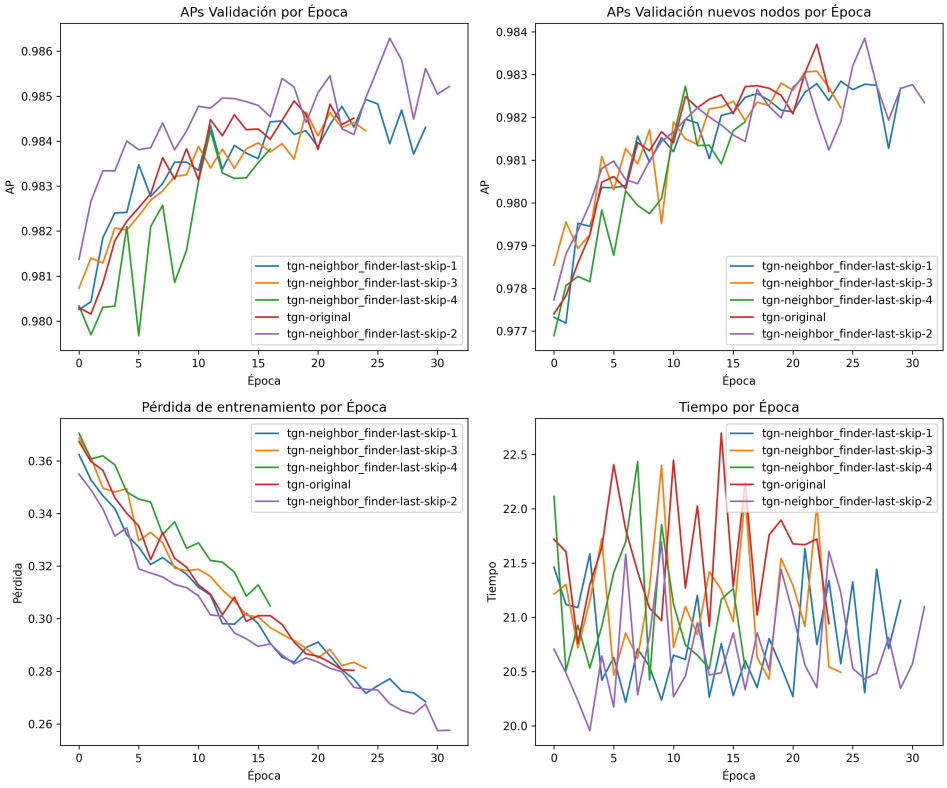



Figura 5.7: Resultados de TGN con buscador de vecinos "last-skip".

- **Adopción del método "last50mean"**: El nuevo modelo propuesto incorpora el método "last50mean" como agregador de mensajes en vez de escoger únicamente el último mensaje de cada nodo.
- **Implementación del buscador de vecinos "last-skip"**: Adicionalmente, el buscador de vecinos del modelo propuesto escoge los nodos mediante el método "last-skip" en lugar de escoger los $n$ últimos nodos vecinos.
- **Utilización de la unidad recurrente cerrada "auto-GRU"**: Por último, debido a la mejora de precisión y de tiempo de ejecución de la unidad recurrente propuesta por Jin et al. (2022), se decidió sustituirla por la GRU del actualizador de la memoria.

Una vez finalizada la selección de las nuevas implementaciones del modelo, se realizaron una serie de pruebas para evaluar el rendimiento tanto del modelo original como del nuevo. Estas pruebas se realizaron en el mismo entorno de trabajo para garantizar la coherencia y fiabilidad de los resultados. La evaluación se realizó utilizando tres conjuntos de datos distintos: Wikipedia, Reddit y Review.

Se eligieron los conjuntos de datos de Wikipedia y Reddit porque forman parte del artículo original en el que se presentó inicialmente el modelo. Ejecutar las nuevas implementaciones en estos conjuntos de datos permite una comparación directa para determinar si se han producido mejoras.

Además, se incluyó el conjunto de datos Review, que forma parte del marco TGB, para probar la eficacia del nuevo modelo en una gama de datos más amplia que

la contemplada en el artículo original. Esta inclusión garantiza una evaluación más completa del rendimiento del modelo en diversos conjuntos de datos.

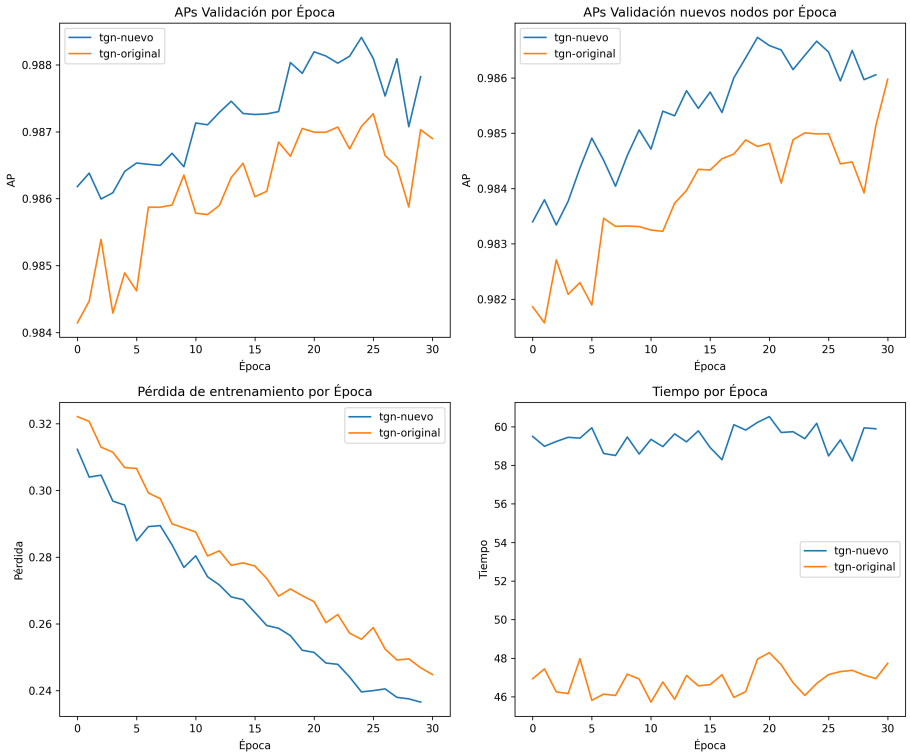



Figura 5.8: Resultados de la predicción de enlaces en Wikipedia para el modelo original y el nuevo.

| | Original | Nuevo |
|---|---|---|
| Test AP | 0.98559 | 0.98697 |
| New Node Test AP | 0.97884 | 0.98078 |

Cuadro 5.7: Resultados de la precisión de test para el modelo original y el nuevo modelo propuesto para el conjunto de datos de Wikipedia.

Como se puede observar en la Figura 5.8 y en la tabla Tabla 5.7, en el conjunto de datos completo de Wikipedia, el nuevo modelo supera durante todo el entrenamiento al original, tanto para la predicción de enlaces con nodos ya vistos como para enlaces con nodos nuevos. Este aumento de la precisión viene acompañado de un ligero aumento del tiempo de ejecución de cada época, proveniente del nuevo agregador de mensajes ”last50mean”.

Por otro lado, la tarea de predicción de enlaces en el conjunto de datos Reddit también muestra una mejora significativa de la precisión, similar a los resultados observados con el conjunto de datos Wikipedia. Esta mejora de la precisión es evidente tanto para los enlaces que incluyen nodos vistos anteriormente como para los que incorporan nodos nuevos. Sin embargo, es importante señalar que la aplicación del nuevo modelo a este conjunto de datos se traduce en un aumento sustancial del tiempo de ejecución. Esta observación sugiere que, a medida que se aumente el tamaño del conjunto de datos, el tiempo de cálculo requerido por el modelo aumentará proporcionalmente.

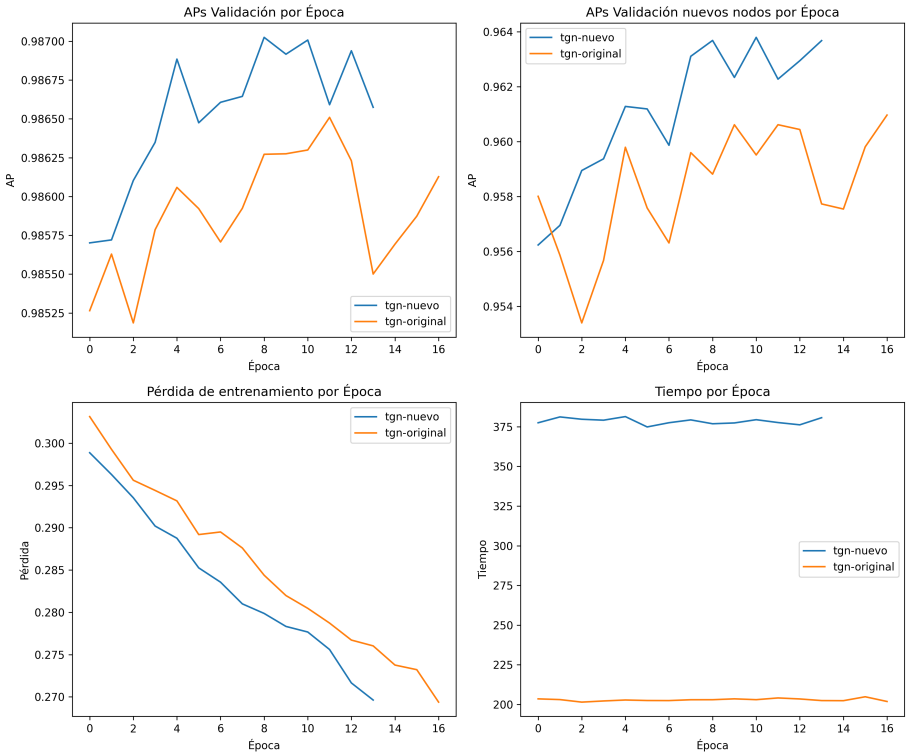



Figura 5.9: Resultados de la predicción de enlaces en Reddit para el modelo original y el nuevo.

| | Original | Nuevo |
|---|---|---|
| Test AP | 0.98739 | 0.98896 |
| New Node Test AP | 0.97627 | 0.97825 |

Cuadro 5.8: Resultados de la precisión de test para el modelo original y el nuevo modelo propuesto para el conjunto de datos de Reddit.

Por último, los resultados del nuevo modelo en el conjunto de datos de Review (Figura 5.10 y Tabla 5.9) también muestran un ligero aumento en la precisión. Sin embargo, el problema del incremento del tiempo de ejecución se vuelve aún más evidente, pasando de un promedio de 7 días a 15 días. Para abordar esta cuestión del aumento del tiempo de ejecución asociado al nuevo modelo, se realizó un experimento para evaluar el rendimiento de distintos valores de $X$ para el agregador de mensajes "lastXmean". Como se detalla en el Capítulo 4, la lógica subyacente a este enfoque es que la reducción del número de mensajes a promediar puede conducir a tiempos de ejecución más cortos. Seleccionando cuidadosamente el valor óptimo de $X$, es posible equilibrar el compromiso entre precisión y eficiencia computacional.

Tras la realización de estas pruebas se llegó a la conclusión de que en casos en los que el conjunto de datos es grande, como el caso del de Review, que contiene 352637 nodos y 4873540 interacciones, es recomendable asignar un valor de $X$ bajo o incluso emplear el método ”last” directamente.

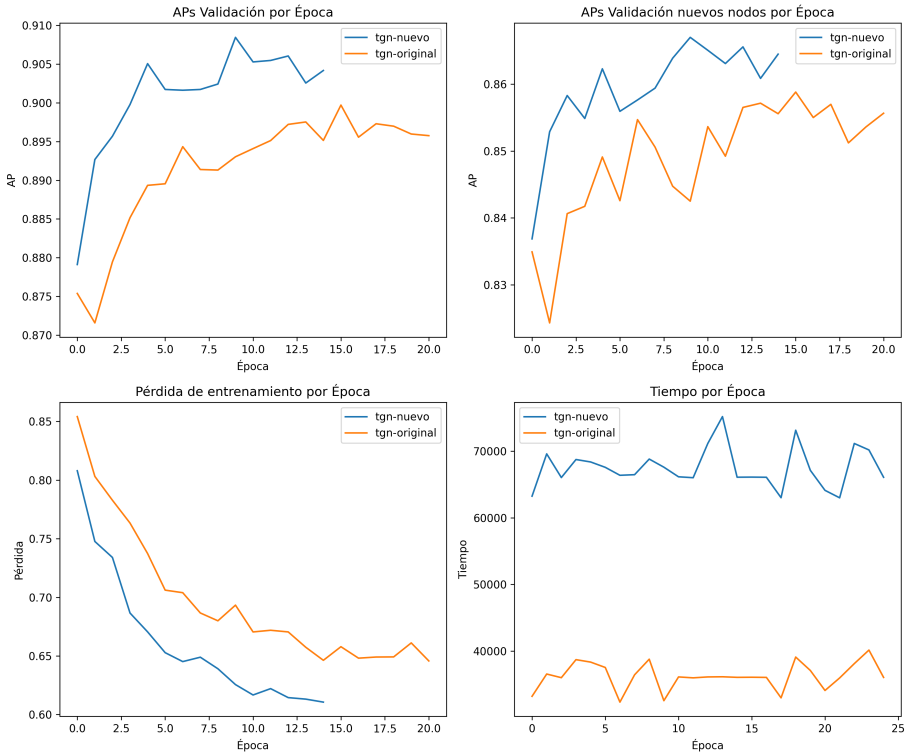



Figura 5.10: Resultados de la predicción de enlaces en Review para el modelo original y el nuevo.

| | Original | Nuevo |
|---|---|---|
| Test AP | 0.90696 | 0.91397 |
| New Node Test AP | 0.85906 | 0.86504 |

Cuadro 5.9: Resultados de la precisión de test para el modelo original y el nuevo modelo propuesto para el conjunto de datos de Review.

## 5.8. Resultados de la clasificación de nodos

Una vez obtenidos los resultados para los conjuntos de datos completos que indican que las nuevas implementaciones mejoran el rendimiento del modelo en la tarea de predicción auto-supervisada de enlaces, es importante comprobar si estas mejoras se trasladan también a la tarea de clasificación de nodos.

Para ello, se evaluó el rendimiento de clasificación de nodos tanto del sistema original como del nuevo utilizando los conjuntos de datos de Wikipedia y Reddit. Esta evaluación se llevó a cabo utilizando los modelos previamente entrenados para la tarea de predicción de enlaces.

| | Original | Nuevo |
|---|---|---|
| Test AP | 0.85955 | 0.87531 |

Cuadro 5.10: Resultados de la precisión de test de clasificación de nodos para el modelo original y el nuevo sobre el conjunto de datos de Wikipedia.

Como se ilustra en las tablas 5.10 y 5.11, la precisión de la clasificación de nodos en

| | Original | Nuevo |
|---|---|---|
| Test AP | 0.61591 | 0.63255 |

Cuadro 5.11: Resultados de la precisión de test de clasificación de nodos para el modelo original y el nuevo sobre el conjunto de datos de Reddit.

los conjuntos de datos de Wikipedia y Reddit muestra un aumento de alrededor del 2 %. Dado que los conjuntos de datos no son excesivamente grandes, la diferencia de tiempo de ejecución entre el modelo original y el nuevo es insignificante y no presenta una desventaja significativa.

# Capítulo 6

# Conclusiones y líneas futuras

Al principio de este trabajo se ha ofrecido una amplia introducción a los grafos, que abarca sus distintos tipos y los problemas de predicción asociados a estos. Además, en un capítulo específico se revisaron los trabajos precedentes relevantes para el estudio principal, con el objetivo de presentar un conocimiento exhaustivo de los métodos de redes neuronales más importantes aplicados a los grafos antes de la investigación actual.

Tras el establecimiento de la base teórica, examinamos a fondo el modelo a estudio propuesto en el trabajo de Rossi et al. (2020). Este análisis tenía como objetivo principal tener una visión detallada del modelo a parte de identificar qué componentes del modelo podrían beneficiarse potencialmente de mejoras y qué componentes eran menos susceptibles de modificación.

Antes de comenzar con el desarrollo de las propuestas, se verificó el correcto funcionamiento del modelo original utilizando los conjuntos de datos Wikipedia, Reddit y Review. La selección de estos tres conjuntos de datos fue deliberada, los dos primeros forman parte de la experimentación del artículo original en el que se introduce el modelo Temporal Graph Network (TGN), mientras que el tercero, Review, es el conjunto de datos más pequeño incluido en el marco Temporal Graph Benchmark (TGB). Aunque podrían utilizarse otros conjuntos de datos de este marco, sus tiempos de ejecución se extenderían a semanas, lo que los hace poco prácticos para este estudio.

Una vez se comprobó el funcionamiento del código proporcionado por los autores del modelo TGN se pasó al desarrollo de nuevas propuestas basadas en otros trabajos relacionados o en el modelo original. Estas modificaciones proponen cambios en los diferentes módulos del modelo y se implementaron dentro del código original. Algunas de estas propuestas son por ejemplo el cambio de la función identidad de los mensajes por un perceptrón multicapa, el uso de diferentes agregadores de mensajes como redes neuronales recurrentes u otros métodos, diferentes buscadores de vecinos e incluso la creación de un módulo adicional de embebido de características.

Tras desarrollar e implementar estas propuestas, se comparó rigurosamente el rendimiento de los nuevos modelos que las incorporaban por separado frente al modelo original para determinar cuáles de ellas resultaban beneficiosas a la hora de aumentar la precisión sin aumentar excesivamente el tiempo de ejecución. Cabe mencionar que para la realización de estas pruebas se creó un conjunto de datos simplificado

a partir del de Wikipedia con el objetivo de disminuir el tiempo de ejecución de los numerosos modelos.

Como se mencionó en el Capítulo 5, algunas de las nuevas implementaciones tuvieron que ser descartadas por su ineficiencia a la hora de aumentar la precisión del modelo y otras debido al gran aumento de tiempo que suponían al modelo. Debido a esto, solo algunas de las propuestas suponen una mejora y por ello no todas se han incluido en el modelo final a probar en los conjuntos de datos completos.

Las tres nuevas propuestas que mejoraban la precisión, tanto para la predicción de enlaces con nodos ya vistos como para enlaces con nodos nuevos, son las tres siguientes:

- **Método "lastXmean" como agregador de mensajes**: La primera de las propuestas que mejora la precisión del modelo es el uso del método "lastXmean" como agregador de mensajes. Este método consiste en la agregación de los últimos $X$ mensajes pertenecientes al mismo nodo mediante la media de estos. De esta manera, los mensajes agregados contienen más información que en el caso de escoger únicamente el último, y se calculan más rápidos que en el caso de hacer la media con todos.
- **Método "last-skip" como buscador de vecinos**: Por otro lado, el método "last-skip" como buscador de vecinos supone también una mejora a la precisión del modelo sin afectar a su tiempo de ejecución. Su funcionamiento se basa en la suposición de que en el contexto de los grafos de interacción, dos nodos comparten mensajes entre ellos de forma continuada, lo que hace posible que si se escogen los últimos nodos como vecinos, se escojan interacciones entre los mismos nodos. Esta elección supone una pérdida de información, por lo que en el método "last-skip" se propone saltar $n$ interacciones por cada una escogida.
- **Unidad recurrente cerrada "auto-GRU" como actualizador de la memoria**: Por último, debido a la mejora de precisión y de tiempo de ejecución de la unidad recurrente propuesta por Jin et al. (2022), se decidió sustituirla por la GRU del actualizador de la memoria.

Como ya se ha mencionado, estas tres mejoras se integraron para formar un modelo TGN mejorado, que luego se comparó con el modelo original utilizando los conjuntos de datos completos de Wikipedia, Reddit y Review. Los resultados demostraron que el nuevo modelo superaba al original en términos de precisión en todos los conjuntos de datos tanto en el entorno transductivo como inductivo. También se pudo observar como para grandes conjuntos de datos con un elevado número de nodos, como el conjunto de datos Review, era más beneficioso emplear un valor bajo para $X$ en el método "lastXmean"del agregador o utilizar directamente el método "last".

En conclusión, las mejoras introducidas en el modelo TGN representan un avance en el campo de las redes gráficas temporales. Al mejorar la capacidad del modelo para predecir enlaces tanto con nodos ya vistos como con nodos nuevos, la investigación aporta valiosas ideas y metodologías para futuros estudios y aplicaciones prácticas. El éxito de estas modificaciones subraya la importancia del perfeccionamiento continuo del desarrollo de modelos de redes neuronales, especialmente en el contexto de conjuntos de datos complejos y dinámicos como los grafos temporales. Este trabajo no sólo avanza en la comprensión teórica, sino que también proporciona herramientas prácticas para un análisis más preciso y eficiente de los grafos temporales.

# Bibliografía


Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, Jiayu Zhou (2017) Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages=65–74.

Shaosheng Cao, Wei Lu, Qiongkai Xu (2016) Deep neural networks for learning graph representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume=30, number=1.

Siheng Chen, Rohan Varma, Aliaksei Sandryhaila, Jelena Kovačević (2015) Discrete signal processing on graphs: Sampling theory. *IEEE Transactions on Signal Processing*, 63(24):6510–6523. IEEE.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Edward De Brouwer, Jaak Simm, Adam Arany, Yves Moreau (2019) GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *Advances in Neural Information Processing Systems*, 32.

Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst (2016) Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Felix A. Gers, Nicol N. Schraudolph, Jürgen Schmidhuber (2002) Learning precise timing with LSTM recurrent networks. *Journal of machine learning research*, 3(Aug), 115–143.

Gori, Marco and Monfardini, Gabriele and Scarselli, Franco (2005) A new model for learning in graph domains. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*, volume=2, pages=729–734. IEEE.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, Reihaneh Rabbany (2024) Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36.

Ming Jin, Yuan-Fang Li, Shirui Pan (2022) Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. *Advances in Neural Information Processing Systems*, 35:19874–19886.

Andrej Karpathy, Li Fei-Fei (2015) Deep visual-semantic alignments for generating

image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages=3128–3137.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, Marcus Brubaker (2019) Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, Pascal Poupart (2020) Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70), 1–73.

Shima Khoshraftar, Aijun An (2024) A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55. ACM New York, NY.

Thomas N. Kipf, Max Welling (2016) Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N. Kipf, Max Welling (2016) Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Srijan Kumar, Xikun Zhang, Jure Leskovec (2019) Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages=1269–1278.

David Liben-Nowell, Jon Kleinberg (2003) The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages=556–559.

Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, Dawei Yin (2020) Streaming graph neural networks. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages=719–728.

James W. Pennebaker, Martha E. Francis, Roger J. Booth (2001) Linguistic inquiry and word count: LIWC 2001. *Mahwah: Lawrence Erlbaum Associates*, 71(2001), 2001.

Liang Qu, Huaisheng Zhu, Qiqi Duan, Yuhui Shi (2020) Continuous-time link prediction via temporal dependent graph neural network. In *Proceedings of the web conference 2020*, pages=3026–3032.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, Michael Bronstein (2020) Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Aliaksei Sandryhaila, José M.F. Moura (2013) Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656. IEEE.

Scarselli, Franco and Gori, Marco and Tsoi, Ah Chung and Hagenbuchner, Markus and Monfardini, Gabriele (2008) The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80. IEEE.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, Xavier Bresson (2018) Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem*

*Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*, pages=362–373. Springer.

Umang Sharan, Jennifer Neville (2008) Temporal-relational classifiers for prediction in evolving domains. In *2008 eighth IEEE international conference on data mining*, pages=540–549. IEEE.

David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, Pierre Vandergheynst (2013) The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98. IEEE.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, Le Song (2017) Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *International Conference on Machine Learning*, pages=3462–3471. PMLR.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha (2019) Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio (2017) Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin (2017) Attention is all you need. *Advances in neural information processing systems*, 30.

Daixin Wang, Peng Cui, Wenwu Zhu (2016) Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages=1225–1234.

Yiwei Wang, Yujun Cai, Yuxuan Liang, Henghui Ding, Changhu Wang, Bryan Hooi (2022) Time-Aware Neighbor Sampling on Temporal Graphs. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, Kannan Achan (2019) Self-attention with functional time representation learning. *Advances in neural information processing systems*, 32.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, Kannan Achan (2020) Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*.

Lin Yao, Luning Wang, Lv Pan, Kai Yao (2016) Link prediction based on common-neighbors for dynamic social network. *Procedia Computer Science*, 83, 82–89. Elsevier.

Bing Yu, Haoteng Yin, Zhanxing Zhu (2017) Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, Wei Wang (2018) Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages=2663–2671.

Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, Aram Galstyan (2016) Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10), 2765–2777. IEEE.

# Anexos

## Anexo A. Simplificación Wikipedia

```python
def simplificar_wikipedia(graph_df, edge_feat):
    '''
    Simplificamos el conjunto de datos de Wikipedia para ahorrar en el tiempo de las pruebas.

    Se eliminan un 50% de los usuarios y las paginas correspondientes, dejando un total de:

        * 4113/8227 usuario y 952/1000 paginas
    '''
    unique_u = graph_df['u'].unique()

    num_nodes_u_to_remove = int(len(unique_u) * 0.5)

    # Establecer una semilla para reproducibilidad
    random.seed(1)

    random_selection = random.sample(list(unique_u), num_nodes_u_to_remove)

    df_filtered = graph_df[graph_df['u'].isin(random_selection)]

    # Reemplazar los valores de 'u' e 'i' con nuevos indices comenzando desde 1
    new_index = {old_index: new_index + 1 for new_index, old_index in enumerate(sorted(set(
        df_filtered['u']).union(df_filtered['i']))))}
    df_reindexed = df_filtered.replace({'u': new_index, 'i': new_index}).sort_values(by='ts')

    new_edge_feat = edge_feat[df_reindexed['idx']]

    # Reemplazar los valores del index e 'idx' con nuevos indices comenzando desde 0
    df_reindexed.reset_index(drop=True, inplace=True)
    df_reindexed.iloc[:, 0]= df_reindexed.index
    df_reindexed['idx'] = df_reindexed.index

    return df_reindexed, new_edge_feat
```