ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

GRAPH NEURAL NETWORK BENCHMARK PER LA SELEZIONE DI CONTENUTO RILEVANTE NELLA LOW-RESOURCE SUMMARIZATION

Elaborato in
Programmazione di Applicazioni Data Intensive

Relatore
Prof. Gianluca Moro

Presentata da
Riccardo Fiorani

Co-relatori
Dott. Lorenzo Valgimigli
Dott. Luca Ragazzi

Terza Sessione di Laurea
Anno Accademico 2022 – 2023

PAROLE CHIAVE

Natural Language Processing

Machine Learning

Deep Neural Networks

Graph Neural Networks

Large Language Model

A chiunque mi sia stato vicino,
e mi abbia aiutato a raggiungere questo traguardo.

Abstract

I filosofi fino ai tempi dell'antica Grecia avevano fatto delle ipotesi sul cervello umano a cui oggi la tecnologia, in particolare i recentissimi studi sui computer danno conferma infatti studiando i computer gli scienziati hanno iniziato a comprendere la reale complessità del cervello e la differenza fra sistemi (e sistemi) complicati e complessi. Che con il termine computer si intendesse arrivare a simulare l'attività mentale umana lo si comprende anche andando ad analizzare l'etimologia e il significato di questa parola: il termine computer è il nome d'agente (cioè il nome derivata da colui che compie un'azione) del verbo inglese "to compute", derivato dal latino computare attraverso il francese computer. L'etimo latina è composto da com = cum (insieme) e putare (tagliare, rendere netto da cui l'odierno potare) e significa propriamente: "confrontare per trarre la somma netta". In Inglese il termine indicava originariamente un essere umano incaricato ad eseguire dei calcoli. Il primo utilizzo nel senso moderno è attestato nel 1897 ma bisognerà attendere la metà degli anni 50 perché questa accezione diventi di uso comune. Il continuo confronto tra cervello umano e computer, che poi in definitiva è una competizione tra natura e tecnologia, sta portando a risultati strabilianti. La simulazione con componenti artificiali e biologici del comportamento del cervello, pur essendo allo stato iniziale porta a risultati importanti come quello dell'intelligenza artificiale. Elemento fondamentale dell'IA è la simulazione del ragionamento. Quest'ultimo utilizza le informazioni, allo stesso modo di come vengono utilizzati dei mattoni per la costruzione di una casa; ma un ammasso di informazioni non costituisce un ragionamento più di quanto un ammasso di mattoni costituiscono una casa. Le origini dell'IA risalgono agli anni '50, quando il matematico e logico Alan Turing pose la domanda "Le macchine possono pensare?" e introdusse il concetto di macchina universale, che è alla base della teoria computazionale moderna. Da allora, l'IA ha attraversato diverse fasi, alternando periodi di grande entusiasmo e avanzamento a periodi di stasi, noti come "inverni dell'IA". Negli ultimi dieci anni la ricerca scientifica in ambito AI ha fatto progressi che gli esperti ritengono superiori alle conoscenze accumulate in precedenza. Questi avanzamenti si sono tradotti in tecnologie software di pubblico dominio che hanno portato allo sviluppo di applicazioni eclatanti in quasi ogni ambito delle discipline

scientifiche. Tuttavia, negli ultimi decenni, grazie ai progressi nell'elaborazione dei dati, all'aumento della potenza computazionale e alla disponibilità di enormi quantità di dati (big data), l'IA ha vissuto una rinascita. Algoritmi come le reti neurali profonde hanno permesso di realizzare progressi significativi in aree come il riconoscimento visivo e vocale, la traduzione automatica e la guida autonoma. Oggi, l'IA è onnipresente e influisce su quasi ogni aspetto della nostra vita quotidiana. Dalle raccomandazioni di prodotti sui siti di e-commerce, alle assistenti virtuali nei nostri smartphone, all'analisi predittiva nella medicina, l'IA sta rivoluzionando settori e creando opportunità inimmaginabili solo pochi anni fa.

Nel cuore di questo elaborato vi è G-SEEK, un approccio innovativo presentato recentemente dal team di ricerca del prof. Gianluca Moro, insieme ai dott. Lorenzo Valgimigli e Luca Ragazzi, in un paper dedicato[1]. G-SEEK è stato introdotto come un modello rivoluzionario per la Summarization, progettato specificamente per affrontare le sfide legate alla Summarization di documenti lunghi, combinando tecniche avanzate di elaborazione del linguaggio naturale con metodi di apprendimento automatico. Questo modello si propone come soluzione per la Summarization abstrattiva basata su grafi, avendo la capacità di creare grafi di conoscenza da grandi quantità di testo, fornendo una rappresentazione semantica ricca e compressa dei dati.

In questo lavoro, sono stati esplorati ulteriormente le potenzialità di G-SEEK combinandolo con nuove Graph Neural Networks(GNN). Ogni GNN porta con sé un insieme unico di caratteristiche e tecniche di modellazione, offrendo diverse prospettive sulla semantica dei grafi creati da G-SEEK.Inoltre, per migliorare ulteriormente l'efficacia, sono stati integrati nuovi Large Language Models(LLM).Ogni LLM, sono noti per le loro capacità di elaborazione del linguaggio naturale, hanno fornito un ulteriore strato di comprensione e hanno migliorato significativamente la qualità dei riassunti generati.

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'era moderna dei dati ha portato con sé una valanga di informazioni, rendendo fondamentale l'importanza dei dataset nel campo della ricerca. Un dataset ben strutturato e curato può fare la differenza tra un modello di machine learning di successo e uno mediocre. Questa esplosione di dati, spesso descritta come "Big Data", riguarda enormi quantità di dati che sono troppo grandi o troppo complessi per essere processati da metodi tradizionali.

Nel contesto del Natural Language Processing (NLP), dove il testo è al centro dell'analisi, la qualità e la pertinenza del dataset diventano ancora più cruciali. L'avvento dei Big Data ha anche catalizzato lo sviluppo di modelli di linguaggio avanzati, come ChatGPT e altri Large Language Models (LLM), che possono processare e generare testo con una precisione senza precedenti. Modelli come questi sono diventati possibili grazie all'immensa quantità di dati testuali disponibili per l'addestramento, segnalando l'importanza dell'era dei Big Data nel plasmare il campo dell'NLP.

1.1 Avanzamenti nell'Intelligenza Artificiale

L'intelligenza artificiale ha fatto passi da gigante negli ultimi decenni, con sviluppi significativi che hanno rivoluzionato il modo in cui percepiamo e utilizziamo la tecnologia. Di seguito è presentata una breve cronologia di alcuni degli eventi più emblematici:

- **1997 - Deep Blue contro Kasparov**: Deep Blue, un computer di scacchi sviluppato da IBM, sconfigge il campione mondiale di scacchi Garry Kasparov. Questo evento è stato un segnale precoce delle potenzialità dell'IA nel superare le capacità umane in compiti specifici.
- **2011 - IBM Watson al Jeopardy!**: Watson, un sistema di IA sviluppato da IBM, vince il quiz televisivo Jeopardy!, sconfiggendo due dei

più grandi campioni del gioco. Questa vittoria ha dimostrato la capacità dell'IA di comprendere e rispondere a domande in linguaggio naturale.

- **2013 - DeepMind e i giochi Atari**: DeepMind Technologies, una startup britannica, sviluppa un algoritmo di apprendimento automatico capace di apprendere a giocare a diversi giochi Atari, superando le prestazioni umane in molti di essi.
- **2016 - DeepMind contro Lee Sedol a Go**: AlphaGo, sviluppato da DeepMind, sconfigge Lee Sedol, uno dei giocatori di Go più forti al mondo. A differenza degli scacchi, il Go è un gioco noto per la sua complessità e profondità strategica, rendendo questa vittoria un risultato significativo per l'IA.
- **2017 - OpenAI e Dota 2**: OpenAI presenta un sistema di IA capace di giocare al videogioco competitivo Dota 2, riuscendo a sconfiggere giocatori professionisti in diverse partite.
- **2020 - ChatGPT da OpenAI**: Basato sull'architettura Transformer, ChatGPT è una delle più grandi e potenti intelligenze artificiali mai create e rilasciate pubblicamente. Questo modello di linguaggio ha impostato nuovi standard nelle applicazioni di NLP, grazie alla sua capacità di generare testo estremamente coerente e contestualmente rilevante, aprendo nuove frontiere per l'interazione uomo-macchina e la generazione automatica di contenuti.

1.2 Big Data: Una Rivoluzione nel Trattamento delle Informazioni

I Big Data rappresentano una delle pietre miliari dell'era digitale contemporanea, costituendo una risorsa inestimabile che ha rivoluzionato il modo in cui le informazioni vengono generate, raccolte, analizzate e utilizzate. Questo termine si riferisce a collezioni di dati così vaste e complesse da risultare difficili da gestire utilizzando metodi tradizionali di elaborazione dei dati.

1.2.1 Caratteristiche dei Big Data

I Big Data sono caratterizzati principalmente da tre dimensioni: volume, varietà e velocità.

- **Volume:** Si riferisce alla quantità massiccia di dati generati ogni secondo da diverse fonti, come i social media, i sensori e gli strumenti di analisi online.

- **Varietà:** I Big Data provengono da una varietà di fonti e possono essere strutturati, semi-strutturati o non strutturati, offrendo una vasta gamma di informazioni utili.

- **Velocità:** Riguarda la frequenza con cui vengono generati nuovi dati e la velocità con cui devono essere processati e analizzati per estrarre informazioni utili.

1.2.2 Analisi dei dati

Nel campo del Natural Language Processing (NLP) e del Text Mining, i Big Data hanno giocato un ruolo determinante, facilitando lo sviluppo e l'addestramento di modelli linguistici avanzati. Grandi volumi di dati testuali disponibili pubblicamente, come articoli, libri, post sui social media e recensioni online, hanno fornito un terreno fertile per addestrare modelli capaci di comprendere, generare e analizzare il linguaggio naturale con una precisione notevole.

Il Text Mining, in particolare, si avvale di queste tecniche per estrarre informazioni utili e conoscenze da grandi quantità di dati testuali non strutturati. Attraverso l'uso di modelli di NLP, è possibile identificare pattern, trend e correlazioni nascoste nei testi, rendendo possibile una vasta gamma di applicazioni, dalla classificazione di documenti all'analisi dei sentimenti e oltre.

1.3 Il Diritto nell'era dell'Intelligenza Artificiale

Nel contesto dell'evoluzione digitale, il trattamento dei Big Data e l'impiego di sistemi di Intelligenza Artificiale (IA) pongono nuove e importanti sfide giuridiche. L'IA, specialmente quando è auto-evolutiva, introduce complessità inedite e una serie di interrogativi che richiedono un'attenta analisi legale e etica.

L'avanzamento dell'Intelligenza Artificiale, con la sua capacità di apprendere autonomamente, solleva questioni giuridiche inedite, in particolare per i sistemi di *self-learning* che pongono sfide significative in termini di spiegabilità e trasparenza. Questa evoluzione solleva interrogativi sulla selezione e sul trattamento dei dati durante la fase di addestramento e sulle implicazioni legali che ne derivano. La necessità di comprendere e spiegare le decisioni prese dalle IA diventa cruciale quando queste influenzano direttamente la vita delle persone, portando al centro del dibattito il diritto degli individui e le linee di responsabilità.

1.3.1 Addestramento e Qualità dei Dati: Punti Chiave e Regolamentazioni

La fase di addestramento è il fondamento su cui si costruisce l'intelligenza di un sistema di IA. È in questo stadio che la qualità e l'integrità dei dati assumono un ruolo cruciale. I dataset devono essere non solo ampi e vari, ma anche accuratamente bilanciati e privi di distorsioni per evitare la creazione di algoritmi che possano, anche involontariamente, perpetuare pregiudizi o discriminazioni.

L'addestramento responsabile di un sistema IA richiede una selezione di dati che rispetti la diversità e la complessità del mondo reale, e che sia conforme ai principi etici e alle normative vigenti. La legislazione, come il GDPR nell'Unione Europea, impone rigidi requisiti sulla raccolta e l'uso dei dati personali, enfatizzando la necessità di trasparenza, il consenso degli utenti e la possibilità per gli individui di comprendere e contestare le decisioni che li riguardano.

Oltre ai principi di base del GDPR, altri regolamenti, come quelli che potrebbero derivare dall'Artificial Intelligence Act, puntano a stabilire linee guida ancora più specifiche per l'addestramento dell'IA. Queste includono la documentazione completa dei processi di addestramento, la revisione periodica dei sistemi per identificare e correggere eventuali bias, e la creazione di meccanismi di auditing indipendenti per garantire la conformità continua alle normative.

1.3.2 L'Importanza della Spiegabilità e dell'Articolo 22 del GDPR

L'Articolo 22 del GDPR (Regolamento Generale sulla Protezione dei Dati) tratta il diritto degli individui di non essere soggetti a una decisione basata unicamente sul trattamento automatizzato, inclusa la profilazione, che produce effetti giuridici che li riguardano o li colpisce in modo significativo. Ciò solleva la necessità di spiegare le decisioni prese dalle IA, soprattutto quando queste decisioni hanno un impatto diretto sulla vita delle persone. Le aziende devono quindi assicurare che, anche nel contesto di sistemi complessi di IA, le decisioni siano interpretabili e giustificabili agli occhi degli utenti e dei regolatori.

1.3.3 L'Artificial Intelligence Act e le Sue Implicazioni

L'Artificial Intelligence Act è una proposta legislativa dell'Unione Europea che mira a regolamentare l'uso dell'IA. Il regolamento propone una classificazione dei sistemi di IA basata sul rischio e stabilisce requisiti chiari per la conformità, concentrandosi su aspetti come la trasparenza, la sicurezza e

i diritti fondamentali. Questo atto rappresenta un passo significativo verso l'istituzione di un quadro giuridico che possa regolare efficacemente l'impiego dell'IA, garantendo la fiducia del pubblico e la protezione degli utenti.

1.4 Contesto applicativo

Il campo del Natural Language Processing (NLP) ha visto una rapida evoluzione negli ultimi anni, con applicazioni che vanno dalla traduzione automatica alla generazione di testo. l'intelligenza artificiale può essere applicata con successo in vari ambiti di ricerca, dal settore medico ai giochi di strategia. Tuttavia, la Summarization di documenti lunghi rimane una delle sfide più complesse e pertinenti nel settore.

1.4.1 L'importanza della Summarization nel settore legale e nei rapporti governativi

La Summarization gioca un ruolo cruciale in vari settori. Nel settore legale, con documenti che spesso superano le duecento pagine, c'è un bisogno crescente di riassunti accurati che possono aiutare avvocati, giudici e altri professionisti a comprendere rapidamente la sostanza di un caso. Allo stesso modo, le agenzie governative producono una quantità enorme di rapporti e documenti. La capacità di estrarre informazioni pertinenti da questi documenti in modo efficiente può influenzare direttamente le decisioni politiche e le strategie di governo.

1.4.2 Articoli di notizie e l'informazione al pubblico

Nell'era dell'informazione, siamo bombardati quotidianamente da una miriade di notizie provenienti da molteplici fonti. Gli articoli di notizie, in particolare, sono uno dei principali mezzi attraverso cui il pubblico si informa sui recenti sviluppi in vari settori, dalla politica alla scienza, dall'economia alle questioni sociali. Tuttavia, il volume crescente di contenuti e la velocità con cui vengono pubblicati possono rendere difficile per i lettori individuare e comprendere i punti chiave di un articolo.

La Summarization di articoli di notizie offre una soluzione a questo problema. Fornendo riassunti concisi degli articoli, si può permettere ai lettori di ottenere rapidamente una panoramica degli eventi chiave, facilitando la comprensione e permettendo una scelta più informata su quali articoli approfondire. Inoltre, per le redazioni e i giornalisti, avere strumenti di Summarization efficaci può aiutare

nella creazione di abstract o anteprime delle notizie, migliorando l'engagement del lettore e aumentando la diffusione di contenuti di qualità.

Nel contesto dei media moderni, dove l'attenzione del lettore è sempre più frammentata e limitata, la Summarization efficace degli articoli di notizie è essenziale per mantenere il pubblico informato e coinvolto.

1.5 Obiettivi

Il team di ricerca del prof. Gianluca Moro, insieme ai dott. Lorenzo Valgimigli e Luca Ragazzi, ha recentemente introdotto un modello innovativo per la Summarization chiamato GSeek [1]. Questo modello è stato progettato per affrontare le sfide legate alla Summarization di documenti lunghi, combinando tecniche avanzate di elaborazione del linguaggio naturale con metodi di apprendimento automatico. I principali contributi di GSeek includono:

- L'utilizzo di una rappresentazione basata su grafi eterogenei del documento, che consente di modellare efficacemente le relazioni tra diverse unità semantiche all'interno del documento.
- L'adozione di tecniche avanzate di estrazione delle informazioni chiave per identificare e fornire frasi salienti a un modello di Summarization abstrattiva.
- Un focus sull'ottimizzazione delle risorse, rendendo G-SEEK particolarmente efficiente in termini di memoria e potenza computazionale.
- Prestazioni notevolmente migliorate in scenari con risorse limitate, come evidenziato dai risultati sperimentali su vari dataset.

L'obiettivo principale di questa tesi, come già sottolineato, è quello di esplorare e dimostrare le capacità del sistema GSeek nel campo della Summarization di documenti lunghi. Nonostante esistano vari strumenti e tecniche per la Summarization, GSeek mira a offrire prestazioni paragonabili ai migliori sistemi attualmente disponibili, ma con un notevole vantaggio: l'utilizzo di risorse significativamente ridotto. Questa ottimizzazione delle risorse non solo rende GSeek una soluzione più sostenibile dal punto di vista economico e ambientale, ma lo rende anche ideale per le applicazioni in tempo reale e per i dispositivi con capacità computazionali limitate.

Inoltre, nel corso di questa ricerca, verranno esaminati diversi approcci basati su Graph Neural Networks (GNN) e Large Language Models (LLM), con particolare attenzione alla loro interazione con GSeek. Si condurrà un'analisi comparativa per identificare le combinazioni più efficaci e per comprendere come

differenti modelli possano influenzare le prestazioni e l'efficienza complessiva del sistema. Si esplorerà inoltre l'applicabilità di questi modelli in domini diversi da quelli originari, al fine di valutare la loro versatilità e adattabilità a diversi contesti e tipologie di dati, estendendo così il potenziale di utilizzo di GSeek oltre i confini del dataset di partenza. Tale esplorazione sarà cruciale per confermare l'elasticità e la robustezza dei modelli in situazioni variegate, garantendo che le soluzioni trovate siano non solo performanti ma anche universalmente applicabili.

Il lavoro di tesi è stato suddiviso nei seguenti capitoli:

- **Capitolo 2** - Panoramica sulle varie tecnologie esistenti in letteratura utilizzabili per il problema posto;
- **Capitolo 3** - Modellazione del progetto;
- **Capitolo 4** - Analisi dei risultati;

Capitolo 2

Le tecnologie disponibili

In questa prima parte verranno descritte più nello specifico tutte le tecnologie che sono state accennate nell'introduzione, per rendere più chiari gli obiettivi e le metodologie impiegate nei capitoli successivi.

2.1 Evoluzione delle Architetture Hardware

Nel contesto della crescente complessità dei modelli di Deep Learning e delle crescenti esigenze computazionali, l'hardware svolge un ruolo cruciale nel determinare la fattibilità e l'efficienza di tali modelli. Le recenti innovazioni hardware, in particolare nell'ambito delle GPU, hanno notevolmente accelerato il progresso nel campo dell'Intelligenza Artificiale.

2.1.1 La prima Legge di Moore

La Prima Legge di Moore, formulata nel 1965 da Gordon Moore, co-fondatore di Intel, postula che il numero di transistor su un microchip raddoppierà approssimativamente ogni due anni, pur mantenendo costanti i costi. Questa osservazione si è tradotta in un aumento esponenziale della potenza di calcolo e della densità di integrazione dei circuiti, con un conseguente decremento del costo per transistor.

Numero di Transistor delle compenenti di un computer La figura 2.1 illustra l'evoluzione del numero di transistor nei processori e la figura 2.2 nelle schede grafiche in relazione all'anno di rilascio confrontando i dati effettivi con le previsioni basate sulla Prima Legge di Moore dimostrando la validità di questa legge empirica nel tempo. Ogni punto blu rappresenta un particolare modello, indicato con il rispettivo nome e il numero di transistor (in milioni). La linea rossa tratteggiata rappresenta la previsione basata sulla Legge di

Moore, che mostra un raddoppio approssimativo del numero di transistor ogni due anni.

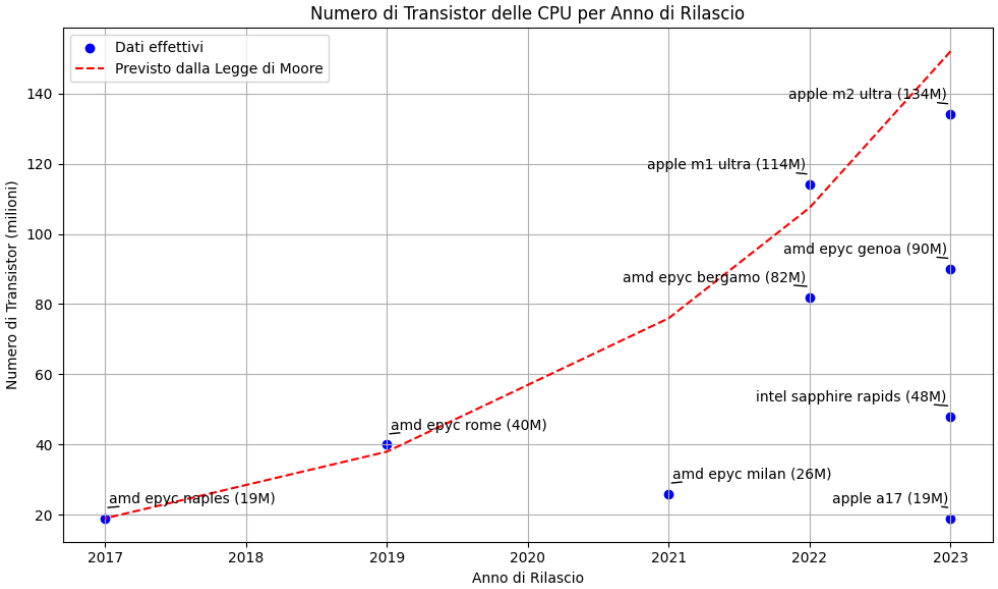

Figura 2.1: Numero di Transistor delle cpu per anno di rilascio.

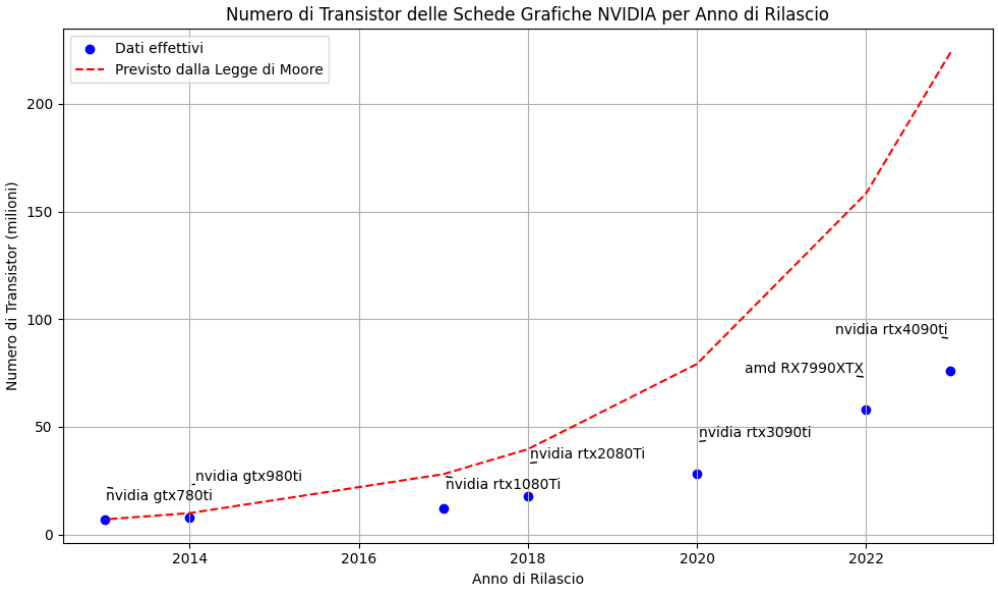

Figura 2.2: Numero di Transistor delle gpu per anno di rilascio.

Tuttavia, la Legge di Moore ha i suoi limiti intrinseci. Alcuni di questi limiti includono:

- Limiti fisici
- Costi di produzione
- Saturazione della frequenza di clock

2.1.2 Limiti Fisici:

Con l'avanzare della miniaturizzazione dei transistor, l'industria dei semiconduttori ha iniziato a confrontarsi con una serie di limiti fisici che minacciano la tradizionale scalabilità dei dispositivi. Questi limiti includono problemi come la corrente di dispersione, la variabilità delle prestazioni e le sfide nella litografia per produrre feature sempre più piccole sui chip. Per superare questi ostacoli, gli ingegneri e i ricercatori hanno iniziato a guardare oltre il tradizionale design bidimensionale dei transistor, esplorando nuove architetture tridimensionali:

Transistor 3D: I transistor 3D, tra cui FinFET e Gate-All-Around (GAA), rappresentano una svolta significativa nella tecnologia dei semiconduttori. Mentre i tradizionali transistor planari hanno limiti in termini di miniaturizzazione e efficienza energetica, le architetture 3D superano questi ostacoli.

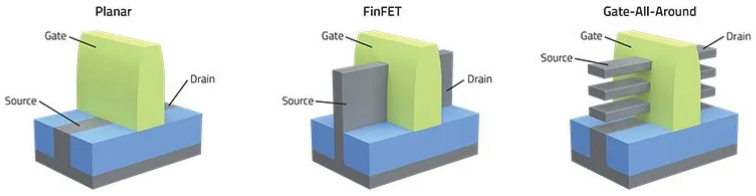

Figura 2.3: Architetture 3d. Fonte [2]

- **FinFET:** Il FinFET è una delle prime architetture 3D adottate su larga scala. Caratterizzato da una struttura a "pinna" rialzata, il FinFET vede il gate avvolgere la pinna su tre lati. Questo design migliora il controllo del flusso di corrente attraverso il canale, riducendo significativamente la corrente di dispersione e consentendo un funzionamento più efficiente a tensioni più basse.
- **Gate-All-Around (GAA):** Il GAA è un'evoluzione del design del FinFET. Invece di avvolgere il gate attorno a una pinna, il GAA lo circonda completamente attorno a un canale cilindrico o

nanofilo. Questo permette un controllo ancora più preciso del flusso di elettroni, portando a una maggiore efficienza energetica e a una riduzione ulteriore dei problemi legati alla miniaturizzazione dei componenti.

L'adozione di questi avanzati design di transistor 3D ha permesso di spingere ulteriormente i limiti della legge di Moore, consentendo la produzione di circuiti sempre più piccoli e potenti, mantenendo al contempo l'efficienza energetica e riducendo l'impatto ambientale.

3D Stacking: Un altro approccio per sfruttare la terza dimensione è impilare diversi strati di dispositivi attivi, come i transistor, uno sopra l'altro. Questo non solo aumenta la densità dei dispositivi, ma permette anche configurazioni innovative, come la cache di memoria posizionata direttamente sopra un core di elaborazione. Un esempio significativo di questa tecnologia è la memoria High Bandwidth Memory (HBM). La HBM è una tecnologia di memoria ad alta larghezza di banda che utilizza il 3D stacking per impilare diversi strati di DRAM in un singolo pacchetto. Questo tipo di memoria è integrata direttamente sullo stesso package del chip, riducendo drasticamente la latenza e aumentando significativamente la larghezza di banda disponibile per il chip, rendendola particolarmente adatta per applicazioni che richiedono trasferimenti di dati ad alta velocità, come le GPU ad alte prestazioni.

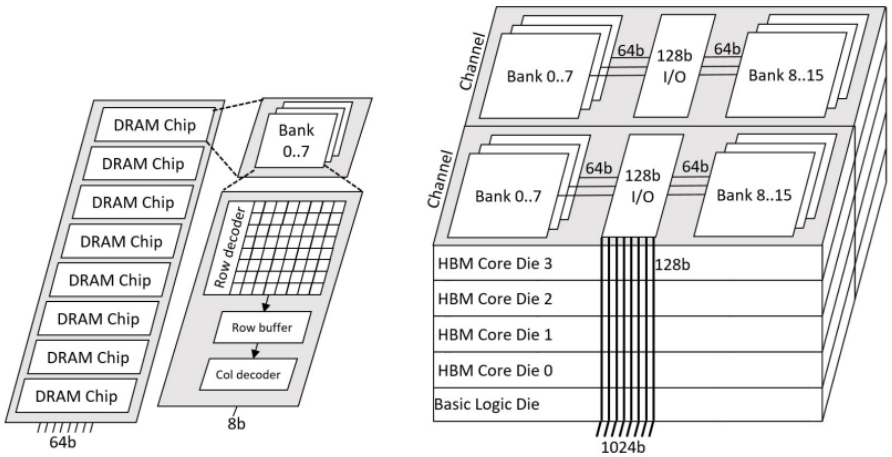

Figura 2.4: Sulla sinstra architettura di una DRAM, a destra di una HBM, Fonte [3]

2.1.3 Costi di Produzione e seconda Legge di Moore:

Di fronte alle sfide fisiche e ai crescenti costi di produzione associati alla continua miniaturizzazione dei transistor, l'industria dei semiconduttori ha cercato nuovi approcci per mantenere vivi i guadagni in performance. Una di queste soluzioni è l'architettura a chiplet di AMD.

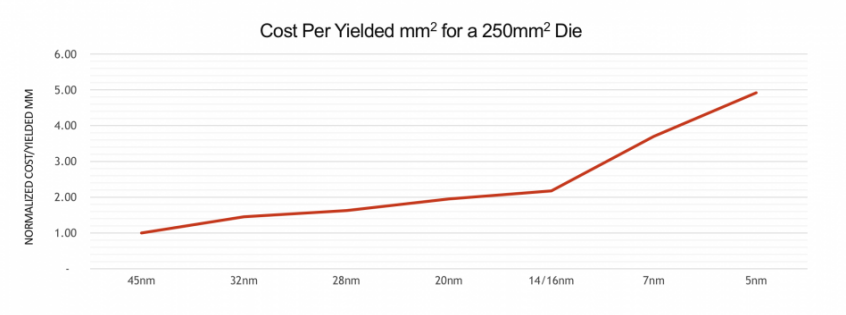

Figura 2.5: Costo normalizzato per chip in funzione del nodo tecnologico. Fonte [4]

Invece di perseguire un singolo die monolitico, sempre più grande e costoso da produrre, l'architettura a chiplet scompone le funzionalità del chip in moduli separati che vengono poi interconnessi. Questo approccio modulare permette una produzione più efficiente, offrendo al contempo flessibilità e scalabilità.

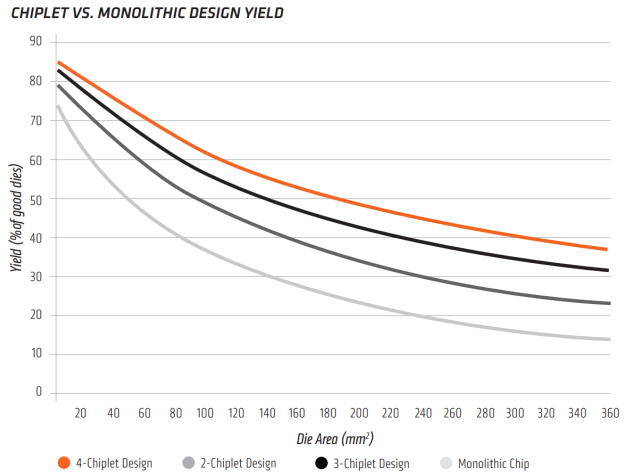

Figura 2.6: Confronto tra le architetture chiplet e monolitici. Fonte [4]

L'adozione di architetture a chiplet può essere vista come una risposta diretta alle sfide poste dalla seconda legge di Moore. Come Moore stesso

ha previsto: «sarebbe molto più economico costruire sistemi su larga scala a partire da funzioni minori, interconnesse separatamente. La disponibilità di varie applicazioni, unita al design e alle modalità di realizzazione, consentirebbe alle società di gestire la produzione più rapidamente e a costi minori». Mentre la miniaturizzazione continua diventa sempre più difficile, la modularità offerta dai chiplets permette all'industria di realizzare guadagni di performance e riduzioni dei costi attraverso innovazioni architetturali.

2.1.4 Saturazione della Frequenza di Clock e Legge di Amdahl

La frequenza di clock di un processore indica la velocità alla quale il processore può eseguire le istruzioni. Per molti anni, un modo diretto per aumentare le prestazioni dei processori era semplicemente aumentare la loro frequenza di clock. Tuttavia, questo approccio ha iniziato a raggiungere i suoi limiti tecnici e termici all'inizio del 21° secolo.

Man mano che la frequenza di clock aumenta, aumenta anche il calore prodotto dal chip. Il dissipatore di calore e le soluzioni di raffreddamento tradizionali diventano insufficienti per gestire l'eccessivo calore prodotto dai processori ad alta frequenza. Si è dovuto passare da sistemi di raffreddamento per convenzione a sistemi di raffreddamento per conduzione. Inoltre, i guadagni di performance derivanti dall'aumento della frequenza di clock hanno iniziato a diminuire a causa dei colli di bottiglia nella comunicazione tra i componenti del processore e della memoria.

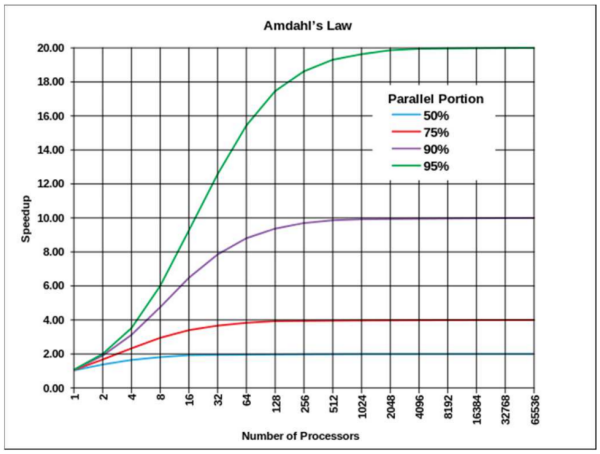

Figura 2.7: Legge di amdahl. Fonte [5]

In questo contesto, la Legge di Amdahl entra in gioco. Formulata da Gene Amdahl nel 1967, questa legge mette in luce i limiti teorici dell'accelerazione delle prestazioni che può essere ottenuta mediante l'elaborazione parallela. La legge afferma che il miglioramento ottenuto da un sistema dipende in gran parte dalla porzione del sistema che può essere parallelizzata. In altre parole, se solo una piccola parte di un'applicazione o di un task può essere eseguita in parallelo, allora ci sono limiti intrinseci ai guadagni di performance che si possono ottenere, indipendentemente dal numero di processori o core disponibili.

Quindi, mentre l'industria ha iniziato a spostarsi verso architetture multi-core per superare i limiti della frequenza di clock, la Legge di Amdahl ci ricorda che non tutte le applicazioni possono beneficiare in modo significativo da tale parallelismo.

2.2 Machine Learning

Il machine learning, ramo dell'intelligenza artificiale, ha rivoluzionato il modo in cui le macchine elaborano e interpretano i dati. Invece di seguire istruzioni predefinite come nella programmazione tradizionale, un sistema di machine learning "impara" dai dati, permettendo a computer e software di diventare più "intelligenti" nel corso del tempo e di adattarsi a nuovi dati senza essere esplicitamente riprogrammati. La sua capacità di apprendere e adattarsi ai dati lo rende un potente strumento per risolvere problemi complessi in una vasta gamma di settori.

2.2.1 Definizione e Principi Fondamentali

Il machine learning può essere definito come l'arte e la scienza dell'allenare i computer a compiere compiti senza una programmazione esplicita. In pratica, si tratta di fornire a un modello grandi quantità di dati e permettergli di apprendere e fare previsioni o prendere decisioni basandosi su di essi.

Il processo di apprendimento inizia con l'inserimento di dati di addestramento nel modello. Questi dati sono processati e analizzati, e il modello "apprende" schemi e tendenze dai dati. Una volta addestrato, il modello può essere utilizzato per fare previsioni su nuovi dati sconosciuti.

2.2.2 Tipologie di Apprendimento

Esistono diverse tipologie di apprendimento nel machine learning, tra cui:

Apprendimento Supervisionato: In questo approccio, l'algoritmo viene addestrato su un set di dati etichettato, il che significa che ogni esempio

nel set di dati è associato a una "risposta corretta". L'obiettivo è di apprendere una mappatura dai dati in ingresso alle uscite desiderate. I modelli supervisionati sono generalmente usati per 2 diversi tipi di problemi:

- **Classificazione** - abbinare l'elemento alla categoria corretta;
- **Regressione** - predire un valore numerico, al posto di una categoria, partendo dalle caratteristiche dell'elemento.

Apprendimento Non Supervisionato: Invece di essere addestrato su un set di dati etichettato, l'algoritmo viene lasciato "libero" di esplorare la struttura e i pattern nei dati da solo. Esempi comuni includono la clusterizzazione e la riduzione della dimensionalità.

Apprendimento per Rinforzo: In questo caso l'algoritmo apprende interagendo con un ambiente e ricevendo feedback sotto forma di ricompense o punizioni. L'obiettivo è di scoprire quale sequenza di azioni porta alla ricompensa massima a lungo termine.

2.2.3 Funzioni di Loss

Le funzioni di perdita svolgono un ruolo cruciale. Queste funzioni quantificano quanto le predizioni del modello siano distanti dai valori effettivamente osservati, fornendo quindi una misura dell'errore commesso dal modello. La scelta della funzione di perdita è dettata dalla natura del problema che si sta affrontando, che può essere di regressione, di classificazione, o altro. Di seguito sono elencate alcune delle funzioni di perdita più comuni utilizzate:

Mean Squared Error (MSE): Utilizzata principalmente per problemi di regressione. La MSE calcola la media dei quadrati delle differenze tra le previsioni $\hat{y}_i$ e i valori reali y_i su un insieme di n esempi:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Cross-Entropy Loss: Adatta per problemi di classificazione, misura la dissimilarità tra la distribuzione prevista p_i e quella reale y_i dei dati. È definita come:

$$H(y, p) = -\sum_{i} y_i \log(p_i)$$

Huber Loss: Una combinazione di MSE e errore assoluto medio (MAE), utilizzata per mitigare l'effetto degli outlier nei dati. Qui, a rappresenta la differenza tra il valore previsto e il valore reale, e δ è un valore soglia:

$$L_\delta(a) = \begin{cases} \frac{1}{2}(a)^2 & \text{for } |a| \leq \delta, \\ \delta \cdot (|a| - \frac{1}{2} \cdot \delta) & \text{otherwise.} \end{cases}$$

Hinge Loss: Utilizzata per le macchine a vettori di supporto (SVM), la Hinge Loss è adatta per la classificazione binaria e si focalizza sull'aumento del margine tra le classi positive e negative. Qui, y_i rappresenta la classe reale e $f(x_i)$ rappresenta il valore previsto:

$$\text{Hinge}(y, f(x)) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))$$

Categorical Crossentropy: Una generalizzazione della Cross-Entropy Loss per problemi di classificazione multiclasse. Misura la dissimilarità tra la distribuzione delle probabilità prevista e quella reale dei dati.

- Binary Cross-Entropy Loss (BCE):
  In questa formula, y_i rappresenta la classe reale e p_i rappresenta la probabilità prevista che un esempio appartenga alla classe positiva:

$$L(y, p) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

- Multiclass Cross-Entropy Loss (CE):
  In questa formula, $y_{i,c}$ indica se la classe reale dell'i-esimo esempio è c, e $p_{i,c}$ è la probabilità prevista che l'i-esimo esempio appartenga alla classe c:

$$L(y, p) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(p_{i,c})$$

L'ottimizzazione delle funzioni di perdita durante la fase di addestramento è cruciale per la costruzione di modelli di machine learning efficaci. La scelta appropriata della funzione di perdita è fondamentale per garantire che il modello apprenda efficacemente i pattern nei dati e adatti i suoi parametri per ridurre al minimo l'errore nelle predizioni.

2.2.4 La Discesa del Gradiente

La Discesa del Gradiente è un algoritmo di ottimizzazione fondamentale nel contesto del machine learning, ed è utilizzato per minimizzare l'errore tra le previsioni del modello e i dati reali, aggiustando iterativamente i parametri del modello. Questo metodo è particolarmente efficace per l'addestramento di modelli in contesti dove la funzione di perdita è differenziabile rispetto ai parametri del modello, come nelle reti neurali.

Principio di Funzionamento

Il principio di funzionamento della Discesa del Gradiente è relativamente semplice. L'algoritmo parte da un punto casuale nello spazio dei parametri e si muove iterativamente verso il punto di minimo locale della funzione di perdita, seguendo la direzione opposta del gradiente della funzione calcolato in quel punto

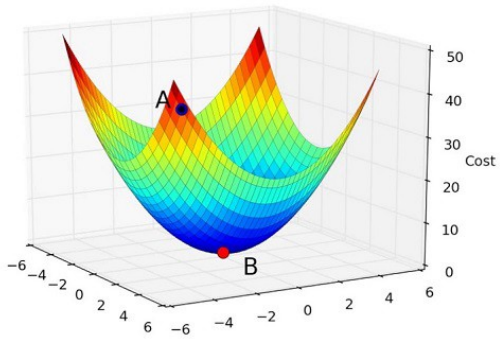

Figura 2.8: Discesa del gradiente dal punto A al punto B visualizzabile graficamente. Fonte [6]

Nel grafico in figura 2.8 ad esempio, le dimensioni x e y che compongono il piano sono quelle relative ai dati di input, mentre la terza dimensione, la z, viene aggiunta per associare l'errore prodotto dal modello.

Per ridurre l'errore quindi, è necessario:

- Calcolare la funzione d'errore e il suo valore attuale;
- Calcolare il gradiente della funzione nel punto, quindi *la direzione* da seguire per raggiungere il punto di minimo;
- Sottrarre al punto iniziale un vettore proporzionale al gradiente;
- Ricominciare da capo finché non viene raggiunto il punto obiettivo.

Questo approccio affonda le proprie radici nel campo della statistica, dove viene chiamato **regressione**. Si occupa di analizzare un insieme di dati che possono essere rappresentati da una funzione (lineare, polinomiale, o di altri gradi) e cercare di approssimare al meglio le variabili dipendenti e indipendenti.

aggiornamento dei parametri

Nel contesto della discesa del gradiente, il processo di aggiornamento dei parametri è guidato dalla formula:

$$\theta_{\text{nuovo}} = \theta_{\text{vecchio}} - \alpha \cdot \nabla J(\theta_{\text{vecchio}})$$

Dove:

- θ_{nuovo}: Rappresenta i parametri aggiornati del modello.
- θ_{vecchio}: Rappresenta i parametri correnti del modello prima dell'aggiornamento.
- α: È il learning rate, un parametro che determina la dimensione del passo nell'aggiornamento dei parametri.
- $\nabla J(\theta_{\text{vecchio}})$: Rappresenta il gradiente della funzione di perdita J rispetto ai parametri θ, calcolato con i parametri correnti θ_{vecchio}.

Il learning rate α gioca un ruolo fondamentale in questo processo, determinando quanto i parametri del modello cambiano ad ogni iterazione. Un α troppo grande potrebbe far sì che il modello non converga, mentre un α troppo piccolo potrebbe rendere il processo di apprendimento eccessivamente lento.

2.2.5 Learning Rate

Il Learning Rate determina la dimensione del passo durante la discesa del gradiente, rappresentando un parametro cruciale nel processo di ottimizzazione. Si tratta di un valore numerico, tipicamente compreso tra 0 e 1, che determina l'ampiezza degli aggiornamenti apportati ai parametri del modello, come i pesi in una rete neurale, durante l'addestramento.

Un learning rate elevato potrebbe causare oscillazioni e impedisce la convergenza del modello al minimo globale della funzione di perdita, mentre un learning rate troppo basso potrebbe rendere il processo di addestramento eccessivamente lento e il modello potrebbe incastrarsi in minimi locali o in punti di sella. Trovare il giusto equilibrio è quindi essenziale per garantire l'efficacia dell'addestramento.

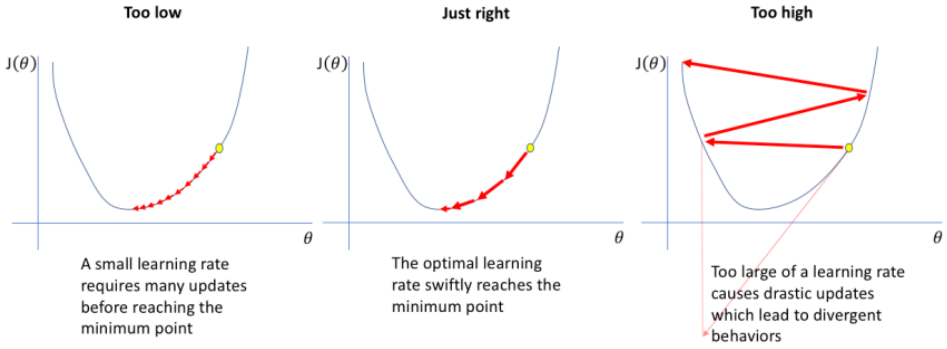

Figura 2.9: Leaning rate. Fonte [7]

Adattamento del Learning Rate

Spesso, si utilizzano strategie per adattare il learning rate nel corso dell'addestramento. Queste strategie sono progettate per ridurre il learning rate man mano che l'addestramento procede, permettendo al modello di avvicinarsi più accuratamente al minimo della funzione di perdita. Alcune tecniche comuni includono:

Step Decay: Riduce il learning rate di un fattore costante ogni numero fissato di epoche.

Exponential Decay: Diminuisce il learning rate a ogni epoca secondo una funzione esponenziale.

1/t Decay: Decresce il learning rate proporzionalmente all'inverso del numero di epoche (t).

Adaptive Learning Rate: Adatta il learning rate in base alla performance del modello, riducendolo quando il miglioramento del modello rallenta o si arresta.

2.2.6 Validazione del Modello

La validazione del modello è un passaggio cruciale nel processo di sviluppo di modelli di machine learning e reti neurali. Questa fase si occupa di verificare le prestazioni del modello, evitando problemi come overfitting o underfitting, e assicurando che il modello sia in grado di generalizzare bene su dati non visti.

Adattamento del modello

L'adattamento del modello è un concetto chiave nella creazione e valutazione di modelli di machine learning e rappresenta il grado con cui un modello apprende e generalizza i pattern presenti nei dati di addestramento. Esistono principalmente tre scenari di adattamento:

- **Overfitting:** Questo scenario si verifica quando un modello apprende i dati di addestramento troppo bene, arrivando a catturare anche il rumore presente nei dati. Un modello in overfitting perde la capacità di generalizzare su dati non visti, risultando in prestazioni scarse su tali dati.

- **Rightfitting:** Il rightfitting rappresenta la situazione ideale. In questo caso, il modello è in grado di apprendere le relazioni sottostanti nei dati di addestramento senza catturare il rumore, mantenendo una buona capacità di generalizzazione su nuovi dati non visti.

- **Underfitting:** Questo termine descrive la situazione in cui un modello non è in grado di apprendere adeguatamente i pattern nei dati di addestramento. Un modello underfitting non cattura le relazioni sottostanti nei dati, risultando in prestazioni scadenti sia sui dati di addestramento che su quelli non visti.

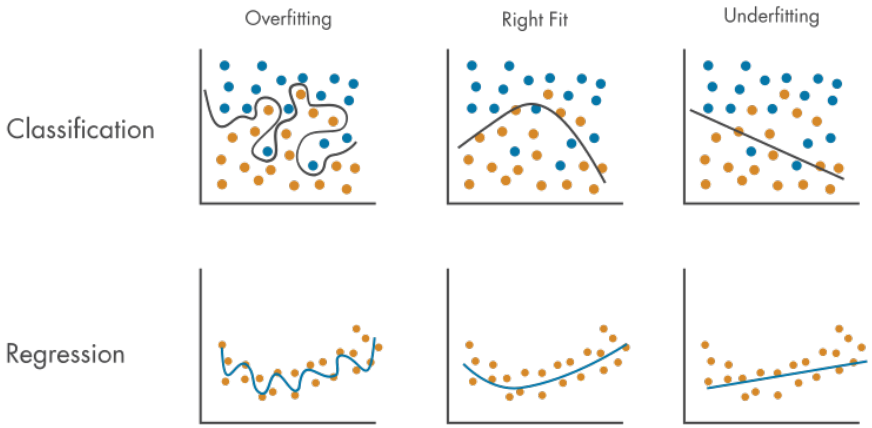

Figura 2.10: Fitting. Fonte [8]

Regolarizzazione

La regolarizzazione è una tecnica cruciale nel machine learning, impiegata per evitare l'overfitting, che può verificarsi quando un modello, eccessivamente complesso, adatta i suoi parametri non solo ai pattern nei dati di addestramento, ma anche al rumore.

Funzione Obiettivo con Termine di Regolarizzazione La regolarizzazione opera aggiungendo un termine di penalità alla funzione obiettivo (o funzione di perdita) che il modello cerca di minimizzare durante l'addestramento. La funzione obiettivo con il termine di regolarizzazione è così definita:

$$L_{\text{reg}}(\theta) = L(\theta) + \lambda R(\theta)$$

dove:

- $L(\theta)$ è la funzione di perdita originale, che misura quanto bene il modello si adatta ai dati di addestramento.
- $R(\theta)$ è il termine di regolarizzazione, che penalizza la complessità del modello.
- λ è un parametro che controlla l'importanza relativa del termine di regolarizzazione rispetto alla funzione di perdita originale.

Tipi di Regolarizzazione

L1 Regolarizzazione (Lasso) La L1 regolarizzazione, o Lasso, aggiunge una penalità alla funzione di perdita proporzionale al valore assoluto dei pesi del modello, inducendo sparsità nei pesi e favorendo modelli con meno parametri non nulli. È utile quando si presume che solo un sottoinsieme delle caratteristiche sia rilevante:

$$R_{\text{L1}}(\theta) = \sum_i |\theta_i|$$

L2 Regolarizzazione (Ridge) La L2 regolarizzazione, o Ridge, penalizza i pesi al quadrato, tendendo a ridurre tutti i pesi verso zero ma senza portarli esattamente a zero, è quindi appropriata quando si credono tutte le caratteristiche siano rilevanti:

$$R_{\text{L2}}(\theta) = \sum_i \theta_i^2$$

Elastic Net L'Elastic Net combina sia la L1 che la L2 regolarizzazione, permettendo di mantenere la selezione delle caratteristiche della L1 e la regolarizzazione uniforme della L2.

Selezione del Parametro di Regolarizzazione Il valore di λ controlla il trade-off tra l'adattamento del modello ai dati e la penalità sulla sua complessità. Un λ troppo alto potrebbe portare a un modello troppo semplice, incapace di apprendere i pattern nei dati (underfitting), mentre un λ troppo basso potrebbe causare overfitting. La scelta di λ è quindi critica e spesso ottenuta tramite tecniche come la cross-validation.

2.3 Neural Network

Le reti neurali rappresentano un'innovazione fondamentale nel campo del machine learning, cercando di emulare il funzionamento dei neuroni nel cervello umano per processare e interpretare le informazioni. Questi sistemi sono composti da neuroni artificiali organizzati in diversi livelli: un livello di input, uno di output e uno o più livelli(da qui il termine "deep learning") intermedi, noti come livelli nascosti (o "hidden"). Esistono due tipologie principali di reti neurali:

Reti Feedforward(es. MLP, CNN): Nelle reti di tipo feedforward, le informazioni si muovono in una sola direzione: dall'input all'output, passando attraverso gli eventuali livelli nascosti. Questo significa che ogni neurone in un livello è collegato ai neuroni del livello successivo, ma non ci sono connessioni che tornano indietro o che collegano neuroni dello stesso livello. Questa è la configurazione più comune per le reti neurali.

Reti Ricorrenti(es.RNN): Al contrario delle reti feedforward, le reti ricorrenti permettono connessioni di feedback. Questo significa che l'informazione può circolare all'interno della rete, creando una sorta di "memoria" interna. Questa caratteristica le rende particolarmente adatte per gestire sequenze di dati, come audio, video o testo. Infatti, in un dato momento temporale una rete ricorrente può accedere alle informazioni processate nei momenti t - 1, t - 2 ecc. Una sottocategoria importante delle reti ricorrenti sono le LSTM (Long Short Term Memory) e le GRU (Gated Recurrent Units), che sono state progettate per affrontare alcune delle sfide associate alla gestione della memoria a breve e lungo termine nelle reti ricorrenti.

2.3.1 Struttura

Come detto, un sistema di deep learning si sviluppa in una struttura molto più complessa e articolata rispetto a un modello di machine learning.

Si creano degli strati di neuroni, elementi fondamentali che altro non sono che dei semplici algoritmi ai quali, dato in input un vettore di n elementi, generano un risultato in output generalmente compreso tra 0 e 1.

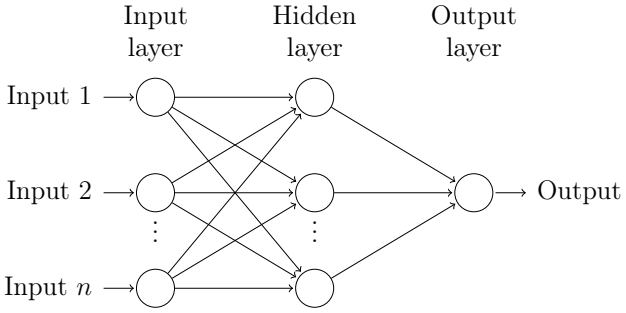

Figura 2.11: Struttura di una rete neurale.

Gli strati sono tra loro collegati, e le informazioni in uscita di uno sono quelle in entrata del successivo. Ogni livello rappresenta un grado di astrazione differente, nel quale vengono gestiti aspetti differenti del dato in input. Nel livello di output viene invece elaborata l'informazione di alto livello richiesta.

Ogni singolo neurone, riceve in input un insieme di valori all'interno di un vettore (detto in certi casi *tensore*) ed elabora internamente il valore y da ritornare in output.

Il neurone non fa altro che eseguire la regressione per minimizzare l'errore, e raggiungere risultati sempre più precisi.

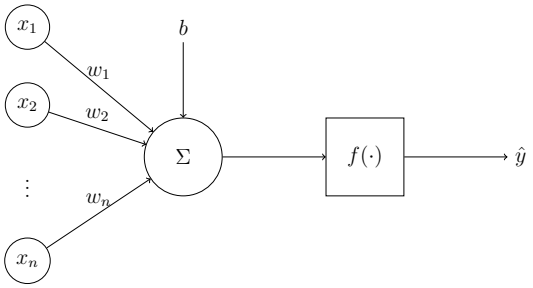

Figura 2.12: Schema di funzionamento unità neuronale.

2.3.2 Feedforward e Backpropagation

Feedforward

Il processo di *feedforward* nelle reti neurali rappresenta la fase in cui l'input viene passato attraverso la rete, da uno strato all'altro, fino a produrre un output. Matematicamente, ogni neurone in uno strato calcola un valore di attivazione basato sull'input ricevuto, i pesi associati, e un bias, e applica poi una funzione di attivazione. Formalmente, l'output y di un neurone è calcolato come:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

dove:

- w_i rappresenta i pesi,
- x_i rappresenta i valori di input,
- b è il bias,
- f è la funzione di attivazione,
- n è il numero di input.

Bias Il bias è un componente fondamentale nella struttura di un neurone. Agisce come un termine di correzione che permette di modificare l'output del neurone, aumentando la flessibilità del modello. Può essere considerato come il termine di intercetta in un'equazione lineare, consentendo la traslazione della funzione di attivazione, cosa che aiuta il modello a adattarsi meglio ai dati.

Backpropagation

La *backpropagation* è il processo mediante il quale l'errore calcolato dall'output della rete viene propagato all'indietro attraverso la rete per aggiornare i pesi e i bias. Questo processo è fondamentale per l'addestramento di una rete neurale.

L'errore è calcolato utilizzando una funzione di perdita L, che misura la differenza tra l'output della rete $\hat{y}$ e il target vero y. L'obiettivo della backpropagation è minimizzare questo errore attraverso l'aggiustamento iterativo dei pesi e dei bias della rete, usando il gradiente dell'errore rispetto a questi parametri.

Formalmente, per un singolo peso w_{ij} che connette il neurone i allo strato l con il neurone j allo strato $l+1$, l'aggiornamento del peso è dato da:

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \alpha \cdot \frac{\partial L}{\partial w_{ij}^{(l)}}$$

dove:

- α è il tasso di apprendimento (learning rate),
- $\frac{\partial L}{\partial w_{ij}^{(l)}}$ è il gradiente della funzione di perdita rispetto al peso $w_{ij}^{(l)}$.

Questo processo di calcolo del gradiente e aggiornamento dei pesi è ripetuto iterativamente per un numero di epoche, o fino a quando l'errore non scende al di sotto di una soglia prefissata.

2.3.3 Funzioni di Attivazione

Le funzioni di attivazione svolgono un ruolo cruciale nelle reti neurali, introducendo non linearità nei modelli. Queste funzioni decidono se un neurone deve essere attivato o no, basandosi sul valore di input ponderato ricevuto. Ecco alcune delle funzioni di attivazione più comuni utilizzate nelle reti neurali:

Sigmoid: La funzione sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$) mappa l'input tra 0 e 1, rendendola utile per i modelli di classificazione binaria. Tuttavia, soffre di scomparsa del gradiente quando l'input è molto grande o molto piccolo.

Tanh: La funzione tangente iperbolica ($\tanh(x) = \frac{2}{1+e^{-2x}} - 1$) mappa l'input tra -1 e 1, offrendo così una gamma più ampia rispetto alla sigmoid. Anch'essa, però, soffre del problema della scomparsa del gradiente.

ReLU: La funzione Rectified Linear Unit (ReLU) definita come $f(x) = \max(0, x)$ è attualmente la funzione di attivazione più popolare nelle reti neurali profonde, principalmente per la sua semplicità computazionale e la capacità di mitigare il problema della scomparsa del gradiente.

Leaky ReLU: Una variante della ReLU, la Leaky ReLU permette un piccolo, non-zero output per valori di input negativi, il che può aiutare a mantenere l'informazione che altrimenti sarebbe persa con la ReLU standard.

Softmax: La funzione Softmax è spesso usata nel livello di output di una rete neurale per la classificazione multiclasse. Converte i punteggi di input in probabilità, con l'output di ogni neurone rappresentante la probabilità che l'input appartenga a una delle classi.

Le funzioni di attivazione sono un elemento chiave per permettere alle reti neurali di modellare e apprendere relazioni complesse nei dati, grazie alla loro capacità di introdurre non-linearità nei modelli.

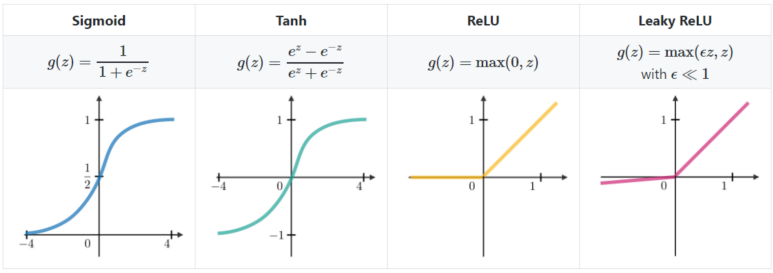

Figura 2.13: Funzioni di attivazione. Fonte [9]

2.3.4 Regolarizzazione Dropout

Il *Dropout* è una tecnica di regolarizzazione specifica per le reti neurali che implica l'eliminazione casuale di alcuni neuroni durante la fase di addestramento. Questo processo previene la co-adattazione dei neuroni e forza la rete a imparare rappresentazioni più robuste e ridondanti dei dati, migliorando così la capacità del modello di generalizzare su dati non visti.

Matematicamente, il Dropout può essere rappresentato come segue:

$$y_i = \begin{cases} 0 & \text{con probabilità } p, \\ \frac{x_i}{1-p} & \text{altrimenti.} \end{cases}$$

Dove:

- y_i è l'output del neurone dopo l'applicazione del Dropout.
- x_i è l'input del neurone.
- p è la probabilità di Dropout, che determina la frazione di neuroni da "spegnere" durante l'addestramento.

Durante la fase di test o valutazione, tutti i neuroni sono attivi, ma i loro output sono scalati in base alla probabilità di Dropout, per mantenere coerenza con la fase di addestramento.

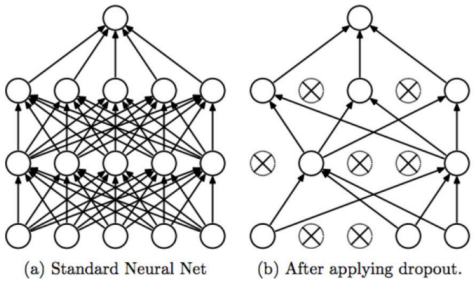

Figura 2.14: Illustrazione della tecnica di Dropout. Fonte [10]

2.3.5 Metriche di Valutazione

Metriche per la Classificazione

In problemi di classificazione, le metriche di valutazione sono utilizzate per misurare la capacità del modello di assegnare correttamente le etichette alle classi. Le principali metriche per la classificazione includono:

Accuracy: L'accuracy è la frazione di predizioni corrette tra il numero totale di input campionati.

$$\text{Accuracy} = \frac{\text{Numero di Predizioni Corrette}}{\text{Numero Totale di Predizioni}}$$

Precision: La precisione è la frazione di predizioni positive che sono effettivamente corrette.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall o Sensitivity: Il recall è la frazione di positivi reali che sono stati identificati correttamente.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score: Lo F1 Score è la media armonica tra precisione e recall, e fornisce un bilanciamento tra essi.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Matrice di Confusione: La matrice di confusione è una tabella che mostra le classificazioni corrette e quelle errate fatte dal modello, evidenziando i falsi positivi, i falsi negativi, i veri positivi e i veri negativi.

Metriche per la Regressione

In problemi di regressione, le metriche di valutazione sono utilizzate per misurare quanto le previsioni del modello si avvicinano ai valori reali. Le principali metriche per la regressione includono:

Errore Quadratico Medio (MSE): Il MSE misura la media dei quadrati delle differenze tra i valori previsti e i valori reali:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Errore Assoluto Medio (MAE): Il MAE misura la media delle differenze assolute tra i valori previsti e i valori reali:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Coefficiente di Determinazione (R-squared): R-squared misura quanto della varianza dei dati può essere spiegato dal modello, con valori più vicini a 1 che indicano una migliore adattabilità del modello ai dati:

$$\text{R-squared} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

K-Fold Cross Validation

La *K-Fold Cross Validation* è una tecnica di validazione incrociata utilizzata per valutare le prestazioni di un modello di machine learning. Questa tecnica è particolarmente utile quando si dispone di un numero limitato di dati, poiché permette di utilizzare l'intero dataset per l'addestramento e la validazione del modello, ottimizzando così l'utilizzo dei dati disponibili.

Procedura: In un K-Fold Cross Validation, il dataset viene diviso in k sottoinsiemi (o "fold") distinti e omogenei. Il modello viene poi addestrato e validato k volte, ogni volta con un fold diverso usato come set di validazione e gli altri $k-1$ fold come set di addestramento. La performance del modello è poi valutata come la media delle performance ottenute nei k cicli di addestramento e validazione.

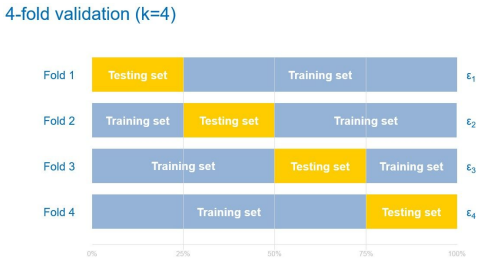

Figura 2.15: Illustrazione del processo di K-Fold Cross Validation. Fonte [11]

Formalizzazione: Sia D il dataset e k il numero di fold. Ogni fold F_i è un sottoinsieme di D tale che:

$$F_i \subset D \quad \text{e} \quad F_i \cap F_j = \emptyset \quad \forall i \neq j$$

Il processo di K-Fold Cross Validation può essere descritto come segue:

1. Dividere il dataset D in k fold disgiunti $F_1, F_2, \ldots, F_k$.
2. Per $i = 1$ a k:
   (a) Usare F_i come set di validazione e $\bigcup_{j \neq i} F_j$ come set di addestramento.
   (b) Addestrare il modello sul set di addestramento e validarlo su F_i.
   (c) Registrare la performance del modello su F_i.
3. Calcolare la performance media del modello sui k fold.

Valutazione: La performance del modello è quindi rappresentata dalla media delle metriche di valutazione (ad es. accuracy, precisione, recall) calcolate su ciascuno dei k fold di validazione:

$$\text{Performance} = \frac{1}{k} \sum_{i=1}^{k} \text{Metrica}_i$$

Questa tecnica permette di ottenere una stima più robusta e affidabile delle prestazioni del modello rispetto a una singola divisione in set di addestramento e validazione, riducendo il rischio di overfitting e offrendo una visione più completa dell'abilità del modello di generalizzare su dati non visti.

2.3.6 Multilayer perceptron (MLP)

Il perceptron, introdotto da Rosenblatt nel 1956, rappresenta uno dei primi modelli di neurone artificiale. Questo modello utilizza una funzione di attivazione lineare a soglia, anche conosciuta come funzione a scalino. Le reti composte da singoli perceptroni o da soli due livelli (input e output) possono essere addestrate attraverso una regola chiamata "delta rule", ispirata alla regola di Hebb. Tuttavia, queste reti a due livelli hanno delle limitazioni: possono apprendere solo funzioni lineari, limitando così il numero di funzioni che possono rappresentare.

Per superare queste limitazioni, è stato introdotto il Multilayer Perceptron (MLP). Un MLP è una rete feedforward che possiede almeno tre livelli, inclusi almeno un livello nascosto, e utilizza funzioni di attivazione non lineari. Questa struttura permette al MLP di apprendere funzioni più complesse. Infatti, un teorema chiamato "universal approximation theorem" stabilisce che un MLP con un solo livello nascosto può approssimare qualsiasi funzione continua che mappa intervalli di numeri reali su un altro intervallo di numeri reali.

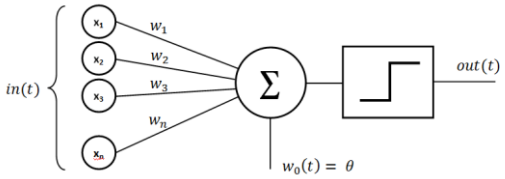

Figura 2.16: Struttura di un neurone per mlp. Fonte [12]

Questa capacità di approssimazione ha reso gli MLP uno degli strumenti prediletti nel campo delle reti neurali per molti anni, almeno fino all'emergere del deep learning. Va notato, però, che pur esistendo teoricamente una soluzione, non è sempre garantito trovare un metodo efficace per ottenerla.

2.3.7 Convolutional Neural Network (CNN)

Le Convolutional Neural Networks (CNN) rappresentano un tipo di rete neurale specializzato, introdotto da LeCun e collaboratori a partire dal 1998. Esse si differenziano dai tradizionali MLP in diversi modi:

Elaborazione Locale: A differenza degli MLP, dove ogni neurone può essere connesso a tutti gli altri neuroni del livello precedente, nelle CNN i neuroni sono connessi solo a una piccola regione locale del livello precedente. Questo concetto di connessione locale garantisce un'elaborazione specifica su porzioni circoscritte dell'input, riducendo notevolmente il numero di connessioni.

Pesi Condivisi: Nelle CNN, i pesi vengono condivisi tra diversi neuroni. Questo significa che neuroni diversi, pur lavorando su differenti porzioni dell'input, utilizzano gli stessi pesi. Ciò comporta una forte riduzione del numero di pesi da addestrare e una capacità di riconoscere pattern specifici indipendentemente dalla loro posizione nell'input.

Alternanza di Livelli: Le CNN alternano livelli dedicati all'estrazione delle caratteristiche (attraverso la convoluzione) a livelli di pooling, che riducono la dimensione dei dati mantenendo le informazioni più rilevanti.

L'architettura di una CNN è specificamente progettata per processare immagini. Sebbene le operazioni di convoluzione, pesi condivisi e pooling semplifichino la rete, esse la rendono anche più efficace nel riconoscimento di pattern nelle immagini rispetto ai modelli fully-connected. Oltre alle immagini, le CNN possono essere adattate per elaborare altri tipi di dati, come segnali audio. L'architettura tipica di una CNN è gerarchica: inizia con un livello di input collegato direttamente ai pixel dell'immagine, segue con vari livelli di convoluzione e pooling, e termina con uno o più livelli fully-connected che operano come un classificatore MLP.

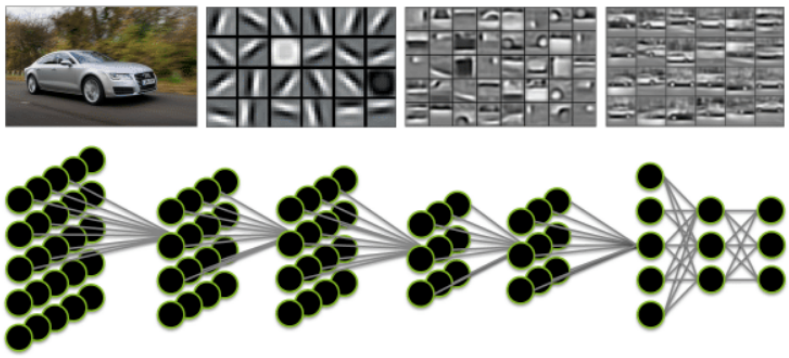

Figura 2.17: Le reti neurali di deep learning hanno una struttura a più livelli. Il Transfer Learning permette di trasferire le caratteristiche apprese tra modelli. Fonte [13]

La Convoluzione è l'operazione fondamentale nelle CNN. Si tratta di applicare un filtro, o maschera, sull'immagine in input. Questo filtro, di dimensioni ridotte rispetto all'immagine, viene fatto scorrere su tutta l'immagine. Ad ogni passo, viene calcolato il prodotto scalare tra i valori del filtro e la porzione di immagine corrispondente, producendo così una nuova immagine, detta "feature map". Questo processo consente di evidenziare determinate caratteristiche dell'immagine, come bordi o angoli, a seconda del filtro utilizzato.

2.3.8 Recurrent Neural Network (RNN)

Le Reti Neurali Ricorrenti (RNN) sono una categoria speciale di reti neurali progettate per lavorare con sequenze di dati. A differenza delle reti feedforward, le RNN hanno connessioni che vanno "all'indietro", cioè dal livello successivo al livello precedente, o connessioni all'interno dello stesso livello. Questa struttura particolare permette alle RNN di mantenere una sorta di "memoria" dei dati processati in precedenza, rendendole particolarmente adatte per l'analisi di sequenze temporali o dati strutturati come le frasi.

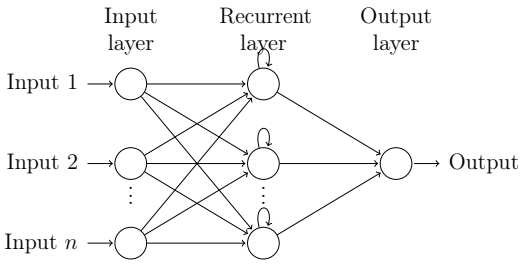

Figura 2.18: Struttura di una rete neurale ricorrente.

In ogni step di una sequenza (ad esempio, in ogni istante temporale t), un'unità in un livello delle RNN riceve non solo l'input corrente x(t) ma anche l'output dello step precedente y(t-1). Ciò consente alla rete di basare le sue decisioni non solo sull'input corrente, ma anche sulla storia degli input passati. Questa "memoria" delle RNN le rende adatte per compiti in cui l'ordine degli input è rilevante, come nel caso dell'elaborazione del linguaggio naturale. Ad esempio, in una frase, non solo le singole parole sono importanti, ma anche la loro sequenza e le relazioni tra di esse.

Struttura di un'unità RNN

Una RNN è composta da unità o celle ricorrenti. Ogni cella RNN prende in input un elemento della sequenza (per esempio, una parola in una frase) e l'output della cella precedente (o uno stato iniziale per la prima cella della sequenza), e produce un output che viene passato alla cella successiva e può anche essere usato come output della rete.

La struttura di base di un'unità RNN può essere descritta dalla seguente formula:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Dove:

- h_t è lo stato nascosto (o output) al tempo t.

- h_{t-1} è lo stato nascosto al tempo $t-1$.
- x_t è l'input al tempo t.
- W_{hh} e W_{xh} sono matrici di pesi.
- b_h è il vettore di bias.
- σ è una funzione di attivazione, come la tangente iperbolica.

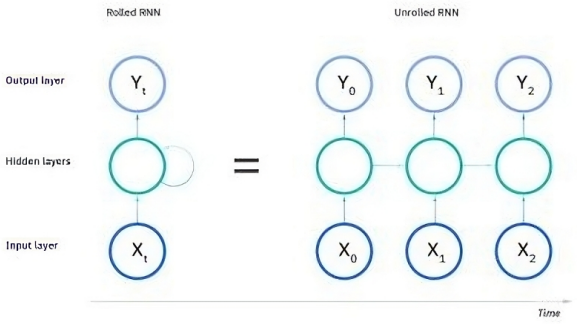

Figura 2.19: Differenza tra la visualizzazione rolled e unrolled di una RNN. Fonte [14]

Long Short-Term Memory (LSTM)

Le Long Short-Term Memory (LSTM) sono un tipo speciale di RNN, sviluppate per mitigare il problema della scomparsa del gradiente, un problema comune nelle RNN standard quando si lavora con sequenze lunghe. Le LSTM sono progettate per conservare le informazioni a lungo termine e sono particolarmente efficaci nel catturare dipendenze a lungo raggio nei dati sequenziali. Una cella LSTM è composta da tre porte principali: la porta di input, la porta di dimenticanza e la porta di output, che lavorano insieme per regolare il flusso di informazioni all'interno della cella.

Gated Recurrent Unit (GRU)

Le Gated Recurrent Unit (GRU) sono un altro tipo di RNN, simili alle LSTM, ma con una struttura leggermente più semplice, che le rende più efficienti computazionalmente. Come le LSTM, anche le GRU sono progettate per catturare dipendenze temporali a lungo termine e mitigare il problema della scomparsa del gradiente. Le GRU hanno due porte principali: la porta di aggiornamento e la porta di reset. Queste porte determinano come le informazioni vengono aggiornate e conservate attraverso gli step temporali.

2.4 Graph Neural Network (GNN)

Le reti neurali su grafi applicano la potenza predittiva del deep learning a strutture dati ricche che rappresentano oggetti e le loro relazioni come punti connessi da linee in un grafo.

Nelle GNN, i punti dati vengono chiamati nodi, collegati da linee, denominate archi, con elementi espressi matematicamente in modo che gli algoritmi di apprendimento automatico possano fare previsioni utili a livello di nodi, archi o interi grafi.

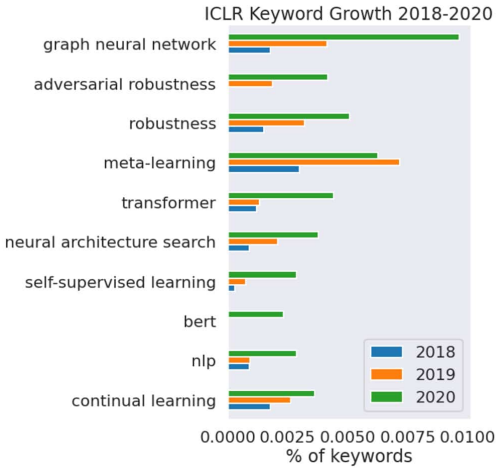

Figura 2.20: Parole chiave di tendenza. Fonte [15]

La figura mostra la crescita delle parole chiave correlate alla Graph Neural Network (GNN) nelle conferenze ICLR (International Conference on Learning Representations) dal 2018 al 2020. Il grafico evidenzia come il termine "graph neural network" sia diventato progressivamente più popolare nel corso degli anni indicati, riflettendo un crescente interesse e focalizzazione su questo argomento nella comunità scientifica dell'apprendimento automatico.

Dal grafico, è evidente che c'è una tendenza crescente nella ricerca relativa alle GNN, suggerendo che le GNN siano diventate una sottoarea importante e in rapida crescita nel campo del deep learning. Questo aumento di popolarità può essere attribuito al potenziale delle GNN di modellare relazioni complesse e strutture di dati non euclidee, che sono essenziali in molti domini applicativi come la bioinformatica, la chimica computazionale e le reti sociali. L'immagine conferma l'emergere delle GNN come un'area chiave di interesse e innovazione nella comunità dell'apprendimento profondo.

2.4.1 Cosa possono fare le GNN

Un numero crescente di aziende sta utilizzando le GNN per migliorare la scoperta di farmaci, la rilevazione delle frodi e i sistemi di raccomandazione. Queste applicazioni e molte altre si basano sulla ricerca di schemi nelle relazioni tra i punti dati.

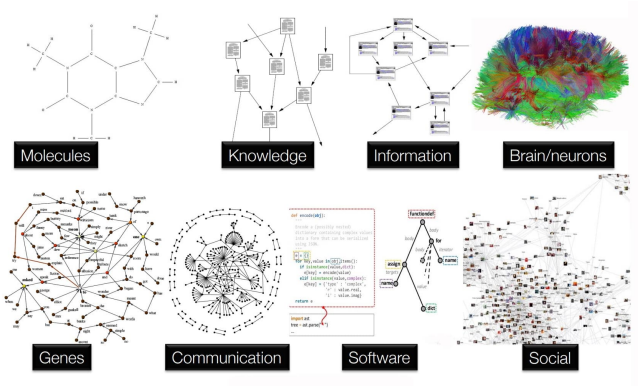

Figura 2.21: Molti campi della scienza e dell'industria può essere espressa sotto forma di grafi. Fonte [15]

I ricercatori stanno esplorando casi d'uso per le GNN in grafica computerizzata, sicurezza informatica, genomica e scienza dei materiali. Un recente studio ha mostrato come le GNN abbiano utilizzato mappe di trasporto come grafi per migliorare le previsioni dell'orario di arrivo.

Molti settori della scienza e dell'industria conservano già dati preziosi nei database di grafi. Con il deep learning, possono addestrare modelli predittivi che scoprono nuove intuizioni dai loro grafi.

2.4.2 Chi usa le reti neurali su grafi

Amazon ha riferito nel 2017 del suo lavoro sull'utilizzo delle GNN per rilevare frodi. Nel 2020, ha lanciato un servizio pubblico di GNN che altri potrebbero utilizzare per la rilevazione delle frodi, sistemi di raccomandazione e altre applicazioni.

Per mantenere l'alto livello di fiducia dei loro clienti, Amazon Search utilizza le GNN per rilevare venditori, acquirenti e prodotti malintenzionati. Utilizzando le GPU NVIDIA, può esplorare grafi con decine di milioni di nodi e centinaia di milioni di archi, riducendo il tempo di addestramento da 24 a cinque ore.

Per quanto riguarda la biopharma, l'azienda GSK mantiene un grafo di conoscenza con quasi 500 miliardi di nodi che viene utilizzato in molti dei suoi modelli di lingua macchina.

2.4.3 Come funzionano le GNN

Ad oggi, il deep learning si è concentrato principalmente su immagini e testi, tipi di dati strutturati che possono essere descritti come sequenze di parole o griglie di pixel. I grafi, al contrario, sono non strutturati. Possono assumere qualsiasi forma o dimensione e contenere qualsiasi tipo di dati, comprese immagini e testo.

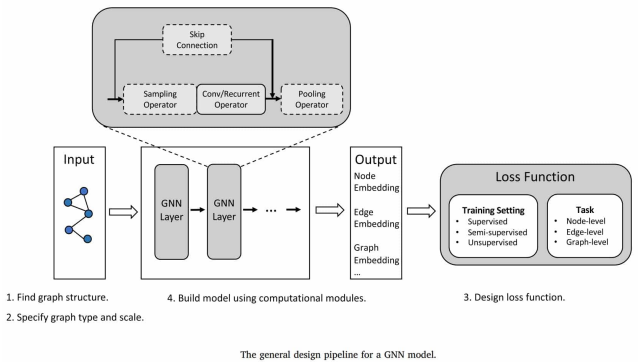

Figura 2.22: Una pipeline GNN ha un grafo come input e previsioni come output. [15]

Utilizzando un processo chiamato "message passing", le GNN organizzano i grafi in modo che gli algoritmi di apprendimento automatico possano utilizzarli.

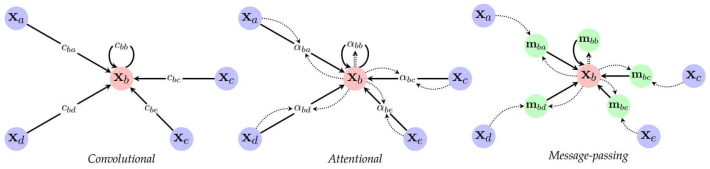

Figura 2.23: Esempi di flussi di dati in tre tipi di GNN. Fonte [15]

2.4.4 Qual è la storia delle GNN

Un articolo del 2009 di ricercatori in Italia è stato il primo a dare alle reti neurali su grafi il loro nome. Ma ci sono voluti otto anni prima che due ricercatori ad Amsterdam dimostrassero la loro potenza con una variante che chiamarono rete neurale convoluzionale su grafo (GCN), che è una delle GNN più popolari oggi. Il lavoro sulla GCN ha ispirato Leskovec e due dei suoi studenti di dottorato di Stanford a creare GraphSage, una GNN che ha mostrato nuovi modi in cui la funzione di "message-passing" potrebbe funzionare.

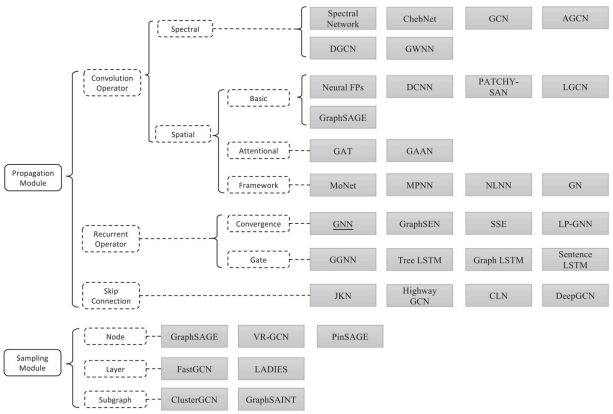

Figura 2.24: Una panoramica delle GNN raffigurato con un albero genealogico delle loro varianti. Fonte [15]

Modellizzazione dei Testi come Grafi

Il testo può essere digitalizzato associando indici ad ogni carattere, parola o token e rappresentando il testo come una sequenza di questi indici. Questo crea un semplice grafo diretto, dove ogni carattere o indice è un nodo e è connesso tramite un arco al nodo che lo segue.

Naturalmente, nella pratica, questo non è di solito il modo in cui il testo e le immagini sono codificati: queste rappresentazioni a grafo sono ridondanti poiché tutte le immagini e tutto il testo avranno strutture molto regolari. Ad esempio, le immagini hanno una struttura a bande nella loro matrice di adiacenza perché tutti i nodi (pixel) sono connessi in una griglia. La matrice di adiacenza per il testo è solo una linea diagonale, perché ogni parola si connette solo alla parola precedente e alla successiva.

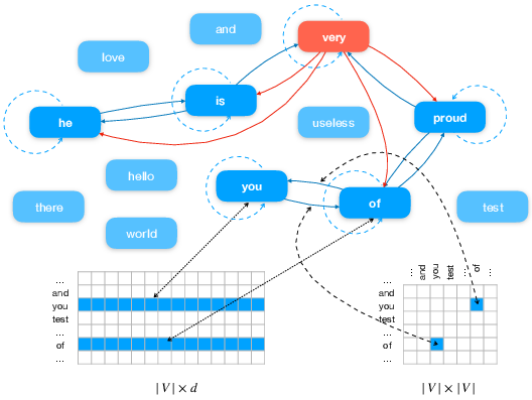

Figura 2.25: Struttura di un testo visto come un grafo. Fonte [16]

2.5 Natural Language Processing (NLP)

L'Elaborazione del Linguaggio Naturale (NLP, dall'inglese Natural Language Processing) è uno dei settori in cui il deep learning ha apportato innovazioni significative, sviluppando tecniche avanzate per l'analisi, la comprensione e la generazione del linguaggio umano. Di seguito vengono illustrate alcune delle principali applicazioni del deep learning nell'NLP:

1. **Classificazione di Frasi:** Determinazione del tono di una recensione, distinzione tra e-mail legittime e spam, verifica della correttezza grammaticale delle frasi e valutazione della relazione logica tra due frasi. Per esempio, può comprendere la classificazione di una recensione come positiva o negativa basandosi sul suo contenuto.

2. **Classificazione di Parole:** Identificazione delle parti del discorso (come nomi, verbi, aggettivi) di ogni parola in una frase, o riconoscere entità denominate come persone, località, e organizzazioni.

3. **Generazione di Testo:** Creazione di contenuto testuale, come completare un prompt con testo generato automaticamente o riempire spazi vuoti in un testo con parole appropriate.

4. **Estrazione di Risposte:** Scenario in cui, dato un contesto e una domanda relativi a quel contesto, il modello deve estrarre la risposta più appropriata basandosi sulle informazioni presenti nel contesto fornito.

5. **Generazione di Nuove Frasi:** focalizzazione sulla creazione di frasi nuove a partire da un input dato, come tradurre un testo in un'altra lingua o generare un riassunto conciso di un testo più lungo.

2.5.1 Pre-processing del Testo per NLP

Il pre-processing del testo è una fase cruciale nell'elaborazione del linguaggio naturale (NLP). Questa fase consiste in una serie di operazioni mirate a pulire e trasformare il testo grezzo in una forma che sia più facilmente analizzabile dai modelli di NLP. Di seguito sono descritti i principali passaggi che compongono il pre-processing del testo:

Pulizia del Testo

La pulizia del testo è un passo cruciale nella maggior parte dei task di NLP. Il processo rimuove o trasforma il testo per eliminare eventuali disturbi o ridondanze, migliorando così l'efficacia dei modelli di NLP. Questo può includere

la rimozione di tag HTML, correzione ortografica, conversione di tutti i testi in minuscolo e la rimozione di caratteri speciali, numeri e punteggiatura.

Tokenizzazione

La tokenizzazione è il processo di suddivisione del testo in unità più piccole, chiamate token, che possono essere parole o simboli. Questo è spesso uno dei primi passi nel pre-processing del testo e serve a preparare il testo per ulteriori analisi, come la lemmatizzazione e la rimozione delle stopwords.

Lemmatizzazione e Stemming

La lemmatizzazione e lo stemming sono tecniche usate per ridurre le parole alla loro forma base o radice. La lemmatizzazione considera il contesto e converte la parola nella sua forma base grammaticale, mentre lo stemming elimina semplicemente gli affissi delle parole, potendo a volte portare a radici che non sono parole valide. Questi processi sono cruciali per ridurre la dimensionalità del testo e migliorare l'efficacia dei modelli di NLP.

Rimozione delle Stopwords

Le stopwords sono parole che appaiono frequentemente nel testo ma non contengono informazioni significative, come "e", "il", "la", ecc. La rimozione delle stopwords è un passo comune nel pre-processing del testo in molte applicazioni di NLP per ridurre la dimensionalità del dataset e migliorare l'efficienza computazionale dei modelli.

Feature Extraction

L'estrazione delle feature è il processo di trasformazione del testo grezzo in un formato che può essere interpretato dai modelli di machine learning. Questo può includere la creazione di vettori di parole (word embeddings), la trasformazione del testo in una rappresentazione del sacchetto di parole (bag-of-words) o l'uso di tecniche come TF-IDF per pesare l'importanza relativa delle parole nel testo.

Word Embeddings

I Word Embeddings sono rappresentazioni vettoriali di parole in uno spazio multidimensionale. Queste rappresentazioni catturano la semantica delle parole e le loro relazioni con altre parole. I modelli come Word2Vec, GloVe e FastText sono usati per creare embeddings di parole che possono essere utilizzati come

input per modelli di deep learning in task di NLP come la classificazione del testo, la traduzione automatica e la generazione di testo.

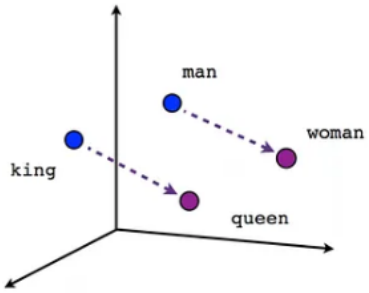

Figura 2.26: Esempio di un sistema di suggerimenti guidato dal machine learning. Fonte [17]

Word2Vec Word2Vec è uno dei modelli più popolari per generare word embeddings. È stato sviluppato da Tomas Mikolov et al. nel 2013, e utilizza reti neurali per imparare rappresentazioni vettoriali di parole da grandi quantità di testo. Word2Vec presenta due architetture principali: CBOW (Continuous Bag of Words) e Skip-gram.

Nell'architettura CBOW, il modello predice la parola corrente basandosi sul contesto (le parole circostanti), mentre nel modello Skip-gram, il processo è inverso: il modello usa la parola corrente per prevedere le parole di contesto. In pratica, Word2Vec è in grado di catturare relazioni semantiche e sintattiche tra le parole, producendo vettori in cui parole semanticamente simili sono vicine nello spazio vettoriale.

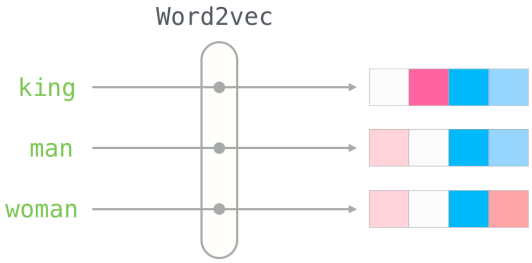

Figura 2.27: Come Word2Vec mappano le parole in uno spazio latente continuo N-dimensionale. Fonte [18]

Word2Vec ha rivoluzionato il campo del Processamento del Linguaggio Naturale, permettendo lo sviluppo di modelli più avanzati e accurati grazie alla sua capacità di catturare sottigliezze semantiche e relazioni tra parole.

Tuttavia, nonostante i suoi successi, Word2Vec ha alcune limitazioni, come l'incapacità di gestire parole con più significati (polisemia) e l'assenza di un modello di memoria a lungo termine.

2.5.2 Summarization

Nel contesto del Natural Language Processing (NLP), la Summarization è la tecnica che si occupa della produzione di un riassunto coerente e conciso di un testo più esteso. L'obiettivo è di mantenere le informazioni chiave del testo originale, riducendo la lunghezza e preservando il significato complessivo. Questa tecnica è estremamente utile in vari ambiti, come l'elaborazione di documenti legali, articoli scientifici, notizie, e molto altro, permettendo una rapida comprensione dei punti principali del contenuto.

Tipologie di Summarization

La Summarization può essere categorizzata principalmente in due tipi:

1. **Extractive Summarization:** In questa forma, il riassunto è creato estraendo le frasi o i segmenti più informativi e rilevanti dal testo originale. Questo metodo non altera il contenuto delle frasi estratte, ma seleziona quelle che meglio rappresentano il significato principale del documento.

2. **Abstractive Summarization:** Questa tipologia genera riassunti creando nuove frasi che rappresentano il contenuto essenziale del testo originale. L'Abstractive Summarization richiede una comprensione più profonda del linguaggio e la capacità di sintetizzare le informazioni, rendendola più complessa rispetto alla forma extractive.

Tecniche e Metodologie

Per realizzare la Summarization, possono essere impiegate diverse tecniche e metodologie, alcune delle quali includono:

1. **Regole e Euristiche:** Si basano sull'utilizzo di regole linguistiche e euristiche per identificare le parti più rilevanti del testo, come la frequenza delle parole, la posizione delle frasi nel documento, ecc.

2. **Machine Learning:** Algoritmi di apprendimento automatico possono essere addestrati per predire l'importanza delle frasi, basandosi su caratteristiche come la presenza di parole chiave, la struttura delle frasi, e così via.

3. **Deep Learning:** Modelli di deep learning, in particolare i modelli di attenzione come i Transformer, hanno dimostrato di essere molto efficaci nella summarization, soprattutto nella forma abstractive, grazie alla loro capacità di comprendere il contesto e generare testo coerente.

Applicazioni e Utilizzo

La Summarization trova applicazione in diversi campi:

1. **Notizie e Media:** Permette di generare riassunti di articoli e notizie, facilitando la fruizione rapida delle informazioni da parte degli utenti.

2. **Ricerca Accademica e Legale:** Aiuta i ricercatori e i professionisti del diritto nella gestione e analisi di grandi volumi di documenti, estraendo le informazioni chiave.

3. **Assistenti Virtuali e Chatbots:** Utilizzata per fornire risposte concise e informazioni rilevanti agli utenti in risposta a domande specifiche.

Sfide

Nonostante i progressi, la Summarization presenta ancora diverse sfide:

1. **Preservazione del Significato:** Creare riassunti che mantengano fedelmente il significato del testo originale è complesso, soprattutto in testi lunghi e complessi.

2. **Coerenza e Leggibilità:** Mantenere coerenza e leggibilità nei riassunti, soprattutto quelli abstractive, è una sfida significativa, dato che i modelli devono generare testo che sia sia informativo che grammaticalmente corretto.

3. **Adattabilità a Diversi Dominii:** Adattare i modelli di summarization a diversi dominii e tipi di testo, come documenti legali o articoli scientifici, richiede un grande sforzo nella creazione di modelli specifici e nella raccolta di dati annotati.

In sintesi, la Summarization è una componente chiave dell'NLP che aiuta nella compressione e comprensione di informazioni testuali, ma presenta ancora molte sfide e opportunità di ricerca e sviluppo.

2.5.3 Sequence-to-Sequence

I modelli Seq2Seq sono modelli di apprendimento profondo che trasformano una sequenza di input (ad esempio, una frase in una lingua) in una sequenza di output (la traduzione della frase in un'altra lingua). Questi modelli sono costituiti da due componenti principali: un encoder e un decoder. L'encoder processa la sequenza di input e crea un vettore di contesto(un vettore di numeri), che è poi utilizzato dal decoder per generare la sequenza di output. In questo contesto, l'uso di reti neurali ricorrenti (RNN) è comune per l'encoder e il decoder, permettendo al modello di gestire sequenze di lunghezze variabili.

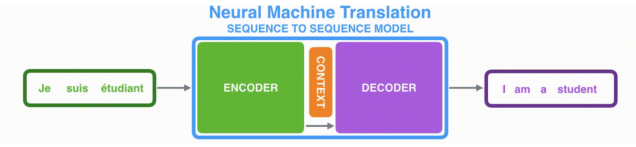

Figura 2.28: Esempio di un sistema di suggerimenti guidato dal machine learning. Fonte [19]

Tuttavia, il vettore di contesto generato dall'encoder si è rivelato essere un collo di bottiglia per i modelli Seq2Seq, specialmente quando si trattava di gestire frasi lunghe. Questo vettore, infatti, doveva contenere tutte le informazioni necessarie della sequenza di input, ma la sua dimensione fissa limitava la quantità di informazioni che poteva contenere, compromettendo la qualità delle traduzioni di frasi lunghe.

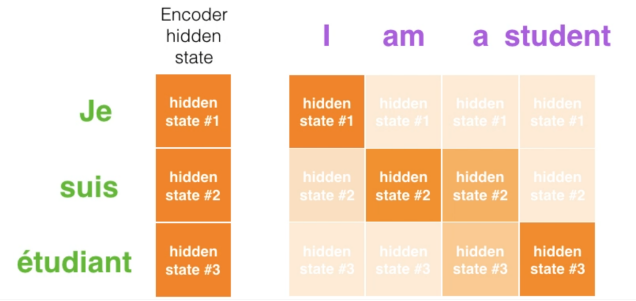

Figura 2.29: Matrice di attenzione visualizzando i pesi di attenzione tra le parole di una frase sorgente e una frase target durante la traduzione con un modello di Seq2Seq con meccanismo di attenzione. Fonte [19]

Per superare questo problema, è stato introdotto il concetto di attenzione. Il meccanismo di attenzione permette al modello di focalizzarsi su differenti parti della sequenza di input mentre genera ogni parola della sequenza di output, invece di affidarsi unicamente su un singolo vettore di contesto. Questo significa

che il decoder ha accesso a tutti gli stati nascosti dell'encoder, e può "prestare attenzione" a parti diverse dell'input ad ogni passo temporale, migliorando notevolmente la capacità del modello di gestire sequenze lunghe e complesse.

- **Encoder:** L'encoder processa ogni parola della sequenza di input, producendo una serie di stati nascosti. Supponiamo di avere una sequenza di input $X = \{x_1, x_2, \ldots, x_T\}$, dove T è la lunghezza della sequenza, e ogni x_i è la rappresentazione vettoriale di una parola. L'encoder calcola gli stati nascosti $H = \{h_1, h_2, \ldots, h_T\}$ come segue:

  $$h_t = f(x_t, h_{t-1})$$

  Dove f è una funzione non lineare, tipicamente un'unità LSTM o una RNN.

- **Calcolo dei Punteggi di Attenzione:** A ogni passo temporale t' del decoder, un punteggio di attenzione è calcolato per ogni stato nascosto h_t dell'encoder, basato sullo stato nascosto attuale del decoder $s_{t'-1}$:

  $$e_{t,t'} = a(s_{t'-1}, h_t)$$

  Dove a è una funzione che calcola il punteggio di attenzione, spesso il prodotto interno:

  $$e_{t,t'} = s_{t'-1}^T \cdot h_t$$

- **Calcolo dei Pesi di Attenzione:** I punteggi di attenzione calcolati $e_{t,t'}$ sono poi trasformati in pesi di attenzione $a_{t,t'}$ usando la funzione softmax:

  $$a_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{i=1}^{T} \exp(e_{i,t'})}$$

- **Calcolo del Context Vector:** I pesi di attenzione $a_{t,t'}$ sono poi utilizzati per calcolare il vettore di contesto $c_{t'}$, che è una somma pesata degli stati nascosti dell'encoder:

  $$c_{t'} = \sum_{i=1}^{T} a_{t,t'} \cdot h_t$$

- **Output e Stato Nascosto:** Il vettore di contesto $c_{t'}$ e lo stato nascosto del decoder $s_{t'-1}$ sono utilizzati per generare l'output $y_{t'}$ del decoder e il nuovo stato nascosto $s_{t'}$:

  $$s_{t'}, y_{t'} = g(s_{t'-1}, c_{t'}, y_{t'-1})$$

  Dove g è una funzione non lineare, solitamente un'altra LSTM o RNN.

In pratica, l'attenzione assegna un punteggio a ciascuna parola della sequenza di input, e questi punteggi determinano quanto ciascuna parola influenzerà l'output a ogni passo temporale. Questo permette al modello di prendere decisioni più informate durante la generazione del testo.

2.6 Transformers

I modelli Transformer rappresentano un'evoluzione significativa nella modellazione delle reti neurali, specialmente nel campo dell'elaborazione del linguaggio naturale (NLP). Introdotto da Google Brain nel 2017, il Transformer ha superato le limitazioni delle reti neurali ricorrenti (RNN) e delle Long Short-Term Memory (LSTM) cells, in particolare nell'addestramento di grandi modelli e nella modellazione di relazioni tra porzioni distanti dell'input.

2.6.1 Architettura

I Transformers sono strutturati in un encoder e un decoder. L'encoder codifica lo stato dell'input, che viene poi passato al decoder. Il decoder opera iterativamente, in modo autoregressivo, producendo un token per volta e riportandolo al suo ingresso. I layer di Attention rappresentano la principale novità in questa architettura, consentendo a ciascun output di essere calcolato come una somma pesata degli input, con i coefficienti di peso che determinano l'importanza relativa di ciascun input.

Encoder:

L'encoder è responsabile della codifica dell'input in una rappresentazione continua che il decoder può utilizzare. È composto da una serie di layer encoder identici. Ogni layer encoder ha due componenti principali:

- **Multi-Head Self Attention Mechanism:** Meccanismo che permette al modello di focalizzare su differenti parti dell'input per ogni parola, assegnando diversi pesi di attenzione a diverse parole. Questo permette al modello di codificare relazioni e dipendenze a lungo raggio tra le parole.

- **Position-wise Feed-Forward Networks:** Dopo il meccanismo di attenzione, la rappresentazione ottenuta passa attraverso una rete neurale feedforward che è applicata separatamente a ogni posizione (parola) dell'input. Questa rete serve a trasformare la rappresentazione di ogni parola in modo non lineare.

Ogni componente è seguita da una normalizzazione rispetto al layer, che serve a stabilizzare l'output di ogni layer prima che passi al successivo.

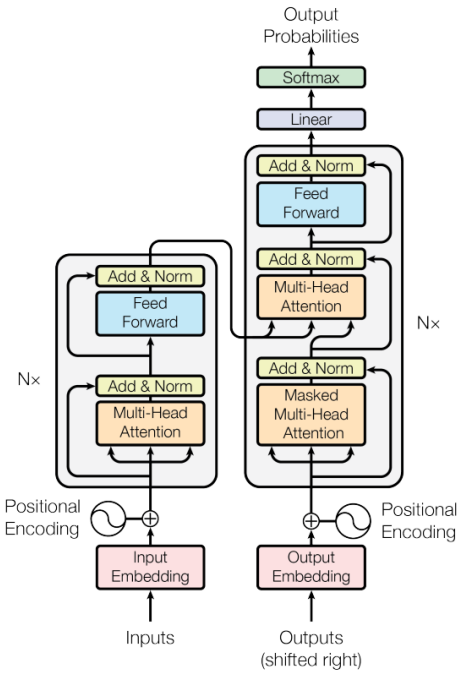

Figura 2.30: Il Transformer segue questa architettura generale utilizzando self-attention impilata e strati completamente connessi punto per punto sia per l'encoder che per il decoder, mostrati rispettivamente nella metà sinistra e destra. Fonte [20]

Decoder:

Il decoder è responsabile della generazione dell'output basandosi sulla rappresentazione continua fornita dall'encoder. È strutturato in modo simile all'encoder, con layer decoder identici composti da tre componenti principali:

- **Masked Multi-Head Self Attention Mechanism:** Simile al meccanismo di attenzione dell'encoder, ma con una maschera che previene l'attenzione alle parole future, permettendo solo l'attenzione alle parole precedenti e alla parola corrente.

- **Multi-Head Attention Mechanism:** Questo layer riceve l'output dell'encoder e permette al decoder di focalizzarsi su differenti parole dell'input durante la generazione dell'output.

- **Position-wise Feed-Forward Networks:** Identico al corrispondente componente dell'encoder, trasforma non linearmente la rappresentazione di ogni parola dell'output.

Anche nel decoder, ogni componente è seguita da una normalizzazione rispetto al layer.

La combinazione di questi meccanismi permette ai modelli Transformer di apprendere rappresentazioni efficaci del linguaggio naturale e di generare sequenze di output coerenti e grammaticalmente corrette.

Layer di Attention

In un layer di attenzione, gli input sono processati attraverso tre componenti principali: Query (Q), Keys (K) e Values (V). Queste componenti sono moltiplicate per delle matrici di pesi prima di essere combinate insieme.

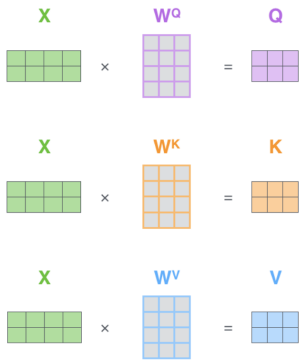

Figura 2.31: Rappresentazione schematica di un layer di attenzione nei modelli Transformer. Fonte [21]

- **Query (Q):** Rappresenta le informazioni che il modello sta cercando. Utilizzato per calcolare i pesi di attenzione.
- **Key (K):** Rappresenta come ogni parola può rispondere a una query. La similarità tra Q e K determina i pesi di attenzione.
- **Value (V):** Rappresenta le informazioni effettive da utilizzare nella costruzione dell'output, pesate in base ai pesi di attenzione calcolati.

Il processo di attribuzione dei pesi è fondamentale: i pesi sono calcolati tramite la misura di similarità tra Query (Q) e Keys (K), permettendo al modello di determinare quali parti dell'input dovrebbero ricevere più attenzione.I pesi di attenzione sono calcolati come segue:

1. **Calcolo del Punteggio di Attenzione:** Il punteggio di attenzione tra ogni parola in input e la query è calcolato tramite il prodotto scalare tra la Query (Q) e le Key (K):

$$\text{score}(Q, K) = Q \cdot K^T$$

2. **Normalizzazione dei Punteggi:** I punteggi di attenzione sono normalizzati dividendo per la radice quadrata della dimensione delle Key:

$$\text{score}_{\text{normalized}}(Q, K) = \frac{Q \cdot K^T}{\sqrt{d_k}}$$

3. **Applicazione della Funzione Softmax:** I punteggi di attenzione normalizzati sono poi trasformati attraverso la funzione softmax per ottenere i pesi di attenzione:

$$\text{weights}(Q, K) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)$$

4. **Calcolo dell'Attenzione:** Infine, i pesi di attenzione sono usati per calcolare l'output dell'attenzione, combinando le Value (V) in base ai pesi di attenzione:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Questo meccanismo di attenzione può essere applicato sia per correlare informazioni di sequenze diverse, sia per stabilire relazioni all'interno della stessa sequenza, un concetto noto come "self-attention".

Il meccanismo di *multi-head attention* estende il concetto di self-attention, permettendo al modello di focalizzare la sua attenzione su differenti posizioni della sequenza di input simultaneamente. Ciò è ottenuto dividendo l'input in diverse "teste" e applicando il meccanismo di self-attention a ciascuna di esse in parallelo. I risultati sono poi concatenati e linearmente trasformati per produrre l'output finale del layer.La formula matematica che rappresenta il meccanismo di multi head attention è la seguente:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

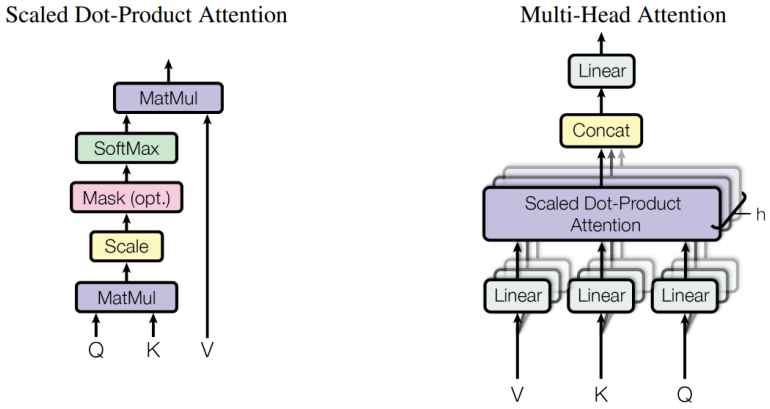

Figura 2.32: (sinistra) Attenzione tramite Prodotto Scalato. (destra) Attenzione Multi-Testa consiste in diversi livelli di attenzione che funzionano in parallelo. [20]

2.6.2 Modelli State-of-the-Art

Nel panorama attuale del Natural Language Processing, diverse architetture all'avanguardia hanno segnato un'evoluzione significativa nel trattamento e nella generazione del linguaggio naturale. In questa sezione, si illustrano alcune delle architetture più influenti, con particolare enfasi sui modelli Transformer, i quali hanno rivoluzionato l'approccio all'apprendimento profondo nel NLP.

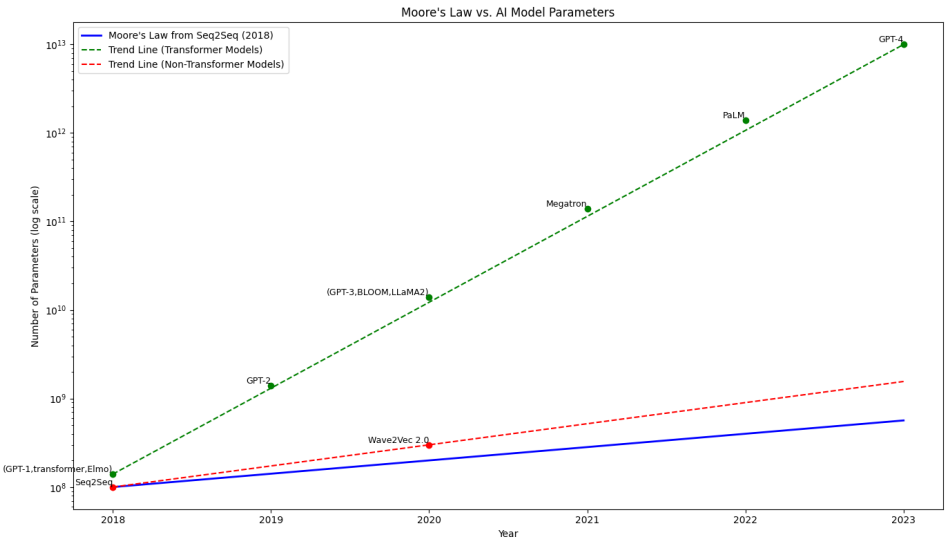

Figura 2.33: Evoluzione dei modelli di NLP e la loro relazione con la Legge di Moore

La figura 2.33 mostra l'evoluzione dei modelli NLP nel tempo, mettendo in luce la rapida crescita del numero di parametri, una tendenza che supera di gran lunga la previsione della Legge di Moore. Quest'ultima, nota per descrivere l'aumento esponenziale del numero di transistor nei microprocessori, trova un parallelo nell'aumento della complessità dei modelli di NLP, suggerendo una nuova "legge" che potrebbe essere definita per il campo dell'intelligenza artificiale.

Capitolo 3

Modellazione del progetto

Nel corso dello sviluppo di questo progetto sono stati condotti numerosi test: alcuni hanno fallito, mentre altri hanno avuto successo. Ogni tecnologia ha portato vantaggi e limiti che sono stati sfruttati in questo lavoro, cercando di ottenere risultati sempre migliori o evidenziando nuovi problemi da affrontare. Questo capitolo contiene tutti gli esperimenti principali effettuati con i relativi risultati e tutti i modelli addestrati.

3.1 Ambiente di sviluppo

Per sviluppare l'intero progetto è stato utilizzato un server fornito dal Professor G. Moro. I modelli utilizzati in questo lavoro sono molto grandi, con milioni di parametri, e il dataset contiene milioni di dati; pertanto, l'ambiente di sviluppo deve avere le risorse necessarie. La soluzione utilizzata nel progetto è spiegata nelle sezioni seguenti.

3.1.1 Server

Il server utilizzato per questo progetto è equipaggiato con una CPU AMD EPYC 7443 e dispone di una memoria RAM di 128 GB. Per gestire operazioni particolarmente onerose e specifiche del campo della grafica e del calcolo parallelo, sono state installate 4 GPU RTX 3090, ciascuna con 24 GB di VRAM. Questa configurazione hardware ha permesso di gestire modelli di apprendimento profondo di grandi dimensioni e di processare dataset molto ampi, fornendo le risorse computazionali necessarie per addestrare i modelli in modo efficace. Questa dotazione di risorse ha rappresentato un elemento chiave per il successo del progetto, permettendo di esplorare e sperimentare con tecnologie e approcci avanzati il campo del machine learning.

CPU: AMD EPYC 7443

La CPU AMD EPYC 7443, utilizzata in questo progetto, è parte della famiglia di processori AMD EPYC di terza generazione e rappresenta una soluzione avanzata e di alto livello, progettata specificamente per gestire carichi di lavoro intensivi e complessi. È particolarmente adatta per applicazioni di calcolo ad alte prestazioni (HPC) e apprendimento automatico.

Caratteristiche Principali:

- **Architettura Chiplet:** La CPU utilizza un design chiplet, che consente di combinare diversi die (chip) su un unico package, migliorando l'efficienza energetica e le prestazioni.
- **3D V-Cache:** Incorpora la tecnologia 3D V-Cache, un innovativo approccio alla cache che permette una maggiore capacità di cache per migliorare le prestazioni nei carichi di lavoro intensivi.

Impatto sul Progetto: In questo progetto, la presenza della CPU AMD EPYC 7443 ha permesso di eseguire efficacemente operazioni di pre-processing dei dati, gestione della memoria e coordinamento delle risorse hardware durante l'addestramento dei modelli di machine learning. La sua capacità di gestire carichi di lavoro complessi e la sua elevata efficienza hanno reso possibile l'esplorazione e l'implementazione di tecniche avanzate di apprendimento automatico, contribuendo significativamente al successo del progetto.

GPU: RTX 3090

Le GPU RTX 3090 installate nel server rappresentano la tecnologia di punta nel settore delle soluzioni grafiche, progettate per fornire prestazioni di livello superiore sia in termini di rendering grafico che di calcolo parallelo.

Caratteristiche Principali:

- **Memoria:** Ciascuna GPU è dotata di 24 GB di VRAM, il che le rende idonee a gestire modelli di grandi dimensioni e a processare ampie quantità di dati in parallelo.
- **Tensor Cores:** I Tensor Cores sono un elemento distintivo. Essi sono progettati per accelerare specificamente le operazioni di prodotto matrice-matrice, che sono centrali in molti algoritmi di deep learning. Ogni Tensor Core è in grado di eseguire operazioni su matrici 4x4 in parallelo, permettendo di effettuare un elevato numero di calcoli simultaneamente.

- **Sparsity Matrix optimization:**I Tensor Cores nelle GPU RTX 3090 sono progettati per sfruttare efficientemente la sparsity nei dati. La sparsity nei modelli di deep learning si riferisce alla presenza di un gran numero di valori zero nei pesi del modello.L'ottimizzazione per la sparsity permette ai Tensor Cores di rilevare e sfruttare schemi di zeri in blocchi contigui all'interno delle matrici, saltando effettivamente le operazioni inutili su valori zero e aumentando così la velocità di elaborazione.

- **Automatic Mixed Precision (AMP) for Deep Learning:** Le GPU supportano la tecnologia Automatic Mixed Precision, che consente di migliorare le prestazioni e ridurre i tempi di addestramento combinando la precisione singola (FP32) e la mezza precisione (FP16) durante le operazioni di calcolo.

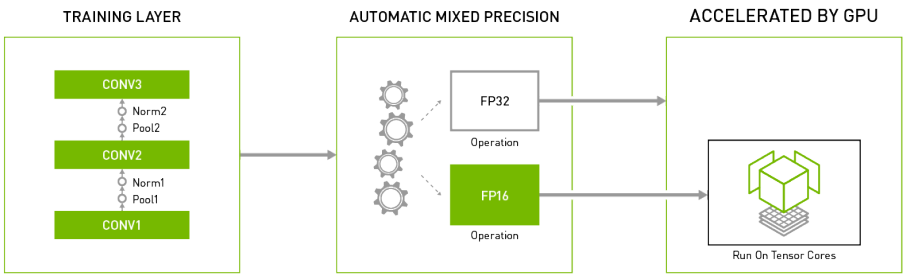

Figura 3.1: schema di un flusso di lavoro in un'architettura di rete neurale. Fonte [22]

Impatto sul Progetto: In questo progetto, l'utilizzo delle GPU RTX 3090 ha permesso una notevole riduzione dei tempi di addestramento dei modelli, facilitando la sperimentazione di diverse configurazioni e ottimizzazioni in tempi ragionevoli. La capacità di queste GPU di gestire operazioni di calcolo ad alta intensità e di parallelizzare i calcoli ha avuto un impatto significativo sullo sviluppo e sulla valutazione dei modelli di apprendimento profondo, consentendo di esplorare e implementare tecniche avanzate di machine learning.

3.1.2 Container

Il container è una tecnologia di virtualizzazione a livello di sistema operativo che permette di eseguire applicazioni e i loro processi in un ambiente isolato e sicuro, separato dal sistema operativo sottostante. A differenza delle macchine virtuali tradizionali, che virtualizzano l'hardware e richiedono un sistema

operativo completo per ogni istanza, i container condividono lo stesso sistema operativo del host ma operano in spazi separati l'uno dall'altro.

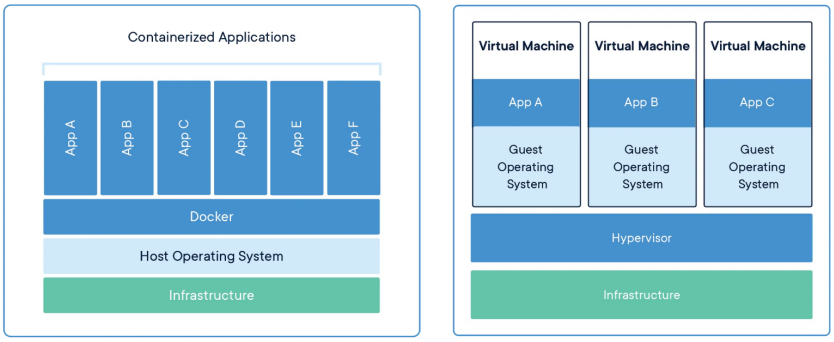

Figura 3.2: Sulla destra dell'immagine è rappresentato il modo in cui le macchine virtuali simulano completamente un server fisico, richiedendo che ogni applicazione abbia una copia del sistema operativo. Sulla sinistra, invece, vediamo varie applicazioni che utilizzano Docker come piattaforma di supporto. Fonte [23]

Ecco alcune caratteristiche chiave dei container:

1. **Leggerezza:** I container sono più leggeri rispetto alle macchine virtuali tradizionali. Poiché non richiedono un sistema operativo separato, consumano meno risorse e possono essere avviati e fermati rapidamente.

2. **Isolamento:** Ogni container funziona come un'entità separata e non interferisce con altri container. Questo isolamento protegge le applicazioni da interferenze e attacchi provenienti da altri container sullo stesso host.

3. **Portabilità:** Poiché un container include tutto ciò di cui ha bisogno per funzionare (codice, librerie, dipendenze), può essere facilmente spostato da un ambiente all'altro (ad esempio, da uno sviluppo locale a un ambiente di produzione in cloud) senza preoccuparsi delle differenze tra i sistemi.

4. **Reproducibilità:** Gli ambienti basati su container sono riproducibili. Ciò significa che se un'applicazione funziona in un container su una macchina, funzionerà allo stesso modo in un altro container su una macchina diversa, purché vengano utilizzate le stesse immagini e configurazioni.

5. **Efficacia nella gestione delle risorse:** I container permettono di sfruttare al massimo le risorse hardware, poiché possono condividere il kernel del sistema operativo host e utilizzare solo le risorse necessarie per eseguire l'applicazione specifica.

Docker

Docker è una piattaforma software che permette di automatizzare il deployment, la scalabilità e l'operatività delle applicazioni all'interno di container. Un container è un'unità software standardizzata che include tutto ciò di cui l'applicazione ha bisogno per funzionare: codice, runtime, librerie di sistema e impostazioni. Questo approccio permette di eseguire l'applicazione in modo coeso e isolato, garantendo la sua portabilità tra diversi sistemi e ambienti.

L'adozione di Docker in questo progetto ha garantito un ambiente di sviluppo riproducibile e isolato, facilitando la gestione delle dipendenze e la configurazione del sistema, e assicurando che l'applicazione funzioni uniformemente su macchine diverse.

Dockerfile Il Dockerfile è un file di testo che contiene una serie di istruzioni usate per creare un'immagine Docker. Ogni istruzione crea un nuovo layer nell'immagine, permettendo di configurare l'ambiente, installare software, e copiare file all'interno del container. Nel Dockerfile per questo progetto è stata utilizzzata come immagine di base 'nvcr.io/nvidia/pytorch:22.12-py3', la quale include un ambiente preconfigurato con PyTorch per l'utilizzo di GPU NVIDIA, ed oltre aìl'imagine di base vengono installati altri pacchetti python che verenna speigati succesivamente. Questo Dockerfile fornisce un metodo efficace e rapido per configurare un ambiente di sviluppo pronto per il progetto, gestendo efficacemente tutte le dipendenze e configurazioni necessarie.

Componenti dell'immagine di base L'immagine nvidia/pytorch:22.12-py3 è un ambiente Docker ottimizzato fornito da NVIDIA che serve come base per applicazioni di deep learning che sfruttano le GPU NVIDIA. Questa immagine include non solo la libreria PyTorch, ma anche una suite di strumenti e librerie progettati per massimizzare le prestazioni dei calcoli su GPU. Ecco una spiegazione di ciascuna delle componenti menzionate:

- **Apex**: Offre tecniche di mixed precision e distributed training per accelerare il training di modelli su GPU.
- **OpenMPI**: Implementazione MPI per comunicazioni efficienti in ambienti paralleli e distribuiti.
- **GDRCopy**: Libreria per accelerare il trasferimento di dati tra GPU e memoria del sistema.
- **NVIDIA TensorRT**: SDK per l'ottimizzazione e l'accelerazione di inferenze di modelli di deep learning su GPU.

- **Torch-TensorRT**: Integrazione di PyTorch con TensorRT per ottimizzare l'inferenza dei modelli.
- **NVIDIA DALI**: Libreria per l'accelerazione della pre-elaborazione dei dati su GPU.
- **MAGMA**: Fornisce algoritmi matematici ottimizzati per GPU per calcoli intensivi.
- **TransformerEngine**: Motore di inferenza per l'ottimizzazione di modelli Transformer su hardware NVIDIA.
- **PyTorch**: Una libreria di machine learning che fornisce massima flessibilità e velocità durante la costruzione e il training di modelli di deep learning, sfruttando al massimo le potenzialità delle GPU NVIDIA.

3.1.3 Python and Frameworks

Python è il linguaggio di programmazione scelto per questo progetto, grazie alla sua sintassi semplice e alla vasta gamma di librerie e framework disponibili, particolarmente adatti per lo sviluppo di soluzioni nel campo del machine learning e del natural language processing.

Nel Dockerfile creato per questo progetto, sono stati installati diversi pacchetti Python, ciascuno con uno scopo specifico:

- `transformers`: Fornisce migliaia di modelli pre-addestrati per il NLP, come BERT e GPT, e facilita la fine-tuning e l'uso di questi modelli.
- `accelerate`: Una libreria per accelerare il training e l'inferenza delle reti neurali su hardware multi-GPU.
- `rouge_score`: Utile per calcolare la metrica ROUGE, comunemente usata per valutare la qualità dei modelli di summarization.
- `wrapt`: Un modulo per la creazione robusta di wrapper di funzioni e metodi in Python.
- `penman`: Una libreria per il parsing e la serializzazione del formato Penman Notation, usato per rappresentare grafici AMR.
- `datasets`: Una libreria per caricare e condividere grandi dataset di NLP.
- `gdown`: Uno strumento per scaricare file da Google Drive.
- `simplejson`: Un semplice encoder e decoder JSON.

- `codecarbon`: Una libreria per misurare l'impronta di carbonio dei workload di calcolo.
- `sentence_transformers`: Permette di calcolare embedding di frasi con modelli basati su BERT.
- `keybert`: Utile per l'estrazione di keyword utilizzando embeddings di parole e tecniche di riduzione della dimensionalità.
- `bert_score`: Una metrica per valutare la qualità delle traduzioni basata su embeddings BERT.
- `wandb`: Weights & Biases, una piattaforma per il monitoraggio e la visualizzazione di esperimenti di machine learning.
- `einops`: Una libreria per la manipolazione di tensori e riduzione di dimensionalità in vari framework di deep learning.
- `torch_geometric`: Una libreria che estende PyTorch per abilitare il deep learning su grafi e altri dati irregolari. Offre moduli flessibili e performanti per la creazione di modelli di Graph Neural Networks (GNNs), prevedendo facilmente la manipolazione e l'analisi di strutture di dati di grafo.

Ogni pacchetto contribuisce a fornire le funzionalità e le risorse necessarie per sviluppare, addestrare, e valutare modelli di machine learning avanzati in questo progetto.

3.1.4 slurm

Slurm è un sistema di gestione di lavori open source, spesso utilizzato per allocare le risorse di calcolo disponibili in un cluster di server tra vari task in esecuzione. È particolarmente utile in ambienti di ricerca e sviluppo dove le risorse di calcolo, come CPU e GPU, sono condivise tra diversi utenti e progetti.

Funzionamento di Slurm

Slurm gestisce l'allocazione delle risorse di un cluster in base a specifiche regole e priorità, permettendo agli utenti di sottomettere job che richiedono una certa quantità di risorse per un determinato periodo di tempo. Una volta sottomesso un job, Slurm si occupa di schedulare l'esecuzione del job, allocando le risorse richieste non appena esse diventano disponibili.

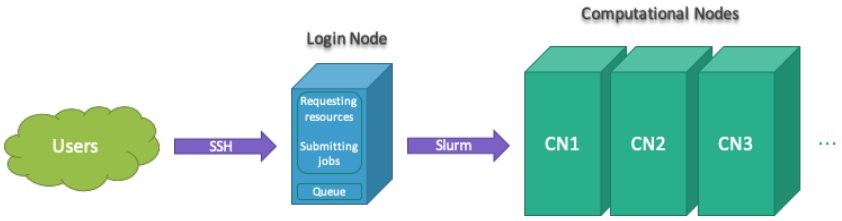

Figura 3.3: Architettura di un sistema di calcolo clusterizzato. [24]

Caratteristiche Principali

- **Scheduling efficiente:** Slurm schedula i lavori in modo efficiente, massimizzando l'utilizzo delle risorse e minimizzando il tempo di attesa.
- **Prioritizzazione dei Lavori:** Gli utenti possono assegnare priorità ai loro lavori, e Slurm allocerà le risorse in base a queste priorità.
- **Gestione delle Risorse:** Slurm permette agli utenti di specificare esattamente quali e quante risorse richiedono, come il numero di nodi, di core CPU, e di GPU.
- **Scalabilità:** Slurm è in grado di gestire cluster di varie dimensioni, da piccoli cluster con pochi nodi a grandi infrastrutture con migliaia di nodi.

Uso in questo Progetto

In questo progetto, Slurm è stato utilizzato per gestire l'allocazione delle risorse del server, permettendo di eseguire esperimenti in parallelo e di sfruttare al meglio le risorse disponibili. L'uso di Slurm ha facilitato l'esecuzione di numerosi esperimenti, aiutando a mantenere un workflow di lavoro organizzato e efficiente.

3.2 Analisi del modello iniziale

Negli ultimi tempi, l'abbondanza crescente di informazioni non strutturate ha promosso soluzioni di Summarization del testo, con l'obiettivo di generare sinossi concise che trasmettono la semantica esatta della Fonte Di recente, grazie agli ultimi progressi nel campo del Natural Language Processing (NLP), la Summarization astrattiva sta ricevendo sempre maggiore attenzione. Questa mira a parafrasare i dettagli più significativi dei documenti invece di limitarsi a recuperarli, come avviene nella Summarization estrattiva. In questo contesto, una sfida particolarmente ardua è l'elaborazione di sequenze massicce, ovvero il compito della Summarization di input lunghi.

Si distingue in particolare tra Summarization di documenti lunghi (LDS) e Summarization multi-documento (MDS). La prima mira a catturare e condensare punti salienti dispersi in un input prolisso (ad es., 10K parole). Nella seconda, la sintesi viene generata da un insieme di fonti correlate per argomento, la cui concatenazione assembla un singolo input lungo, consentendo di affrontare il compito di Summarization come nell'LDS.

Le soluzioni attualmente più avanzate per la Summarization di input lunghi si basano sui transformer, che sono in grado di catturare in modo efficace le relazioni a lunga distanza tra le parole con il meccanismo di self-attention. Tuttavia, questi modelli sono caratterizzati da una limitazione strutturale che collega proporzionalmente il loro utilizzo di memoria alla dimensione dell'input, rendendoli eccessivamente esigenti in termini di risorse quando si elaborano testi lunghi.

Un approccio promettente per attenuare questi ostacoli consiste nell'utilizzare un grafo semantico per rappresentare l'input. Intuitivamente, aggregando tutte le informazioni della fonte, è possibile individuare ed estrarre le frasi degne di nota, evitando la classica troncatura dell'input e fornendo ai modelli più campioni di alta qualità che li aiutino a imparare più rapidamente in condizioni di scarsità di dati.

Tra le proposte recenti in questo contesto, Il team di ricerca del prof. Gianluca Moro, insieme ai dott. Lorenzo Valgimigli e Luca Ragazzi, ha recentemente introdotto un modello innovativo per la Summarization chiamato GSeek, un approccio di Summarization basato su grafi. Utilizzando un grafo eterogeneo per rappresentare l'input, G-SEEK estrae e fornisce le frasi più rilevanti a un sommarizzatore astrattivo per la generazione della sintesi.

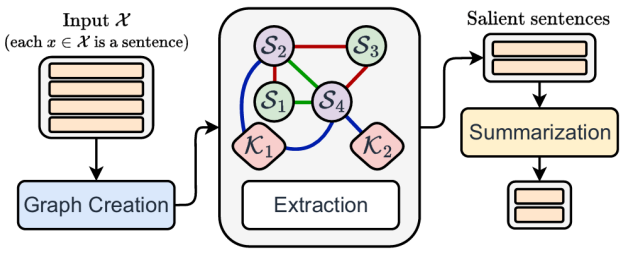

Figura 3.4: Panoramica dell'approccio G-SEEK. Un input lungo X (ogni x in X è una frase) viene convertito in un grafo eterogeneo: i diamanti rosa rappresentano i nodi delle parole chiave K, i cerchi viola denotano i nodi delle frasi S contenenti parole chiave, i cerchi verdi indicano il contesto di S, e i diversi segmenti tra i nodi simboleggiano gli archi. Le frasi salienti vengono estratte e fornite a un PLM generativo per produrre il riassunto. Fonte [1]

3.2.1 Soft Labeling

I recenti progressi nel campo del Natural Language Processing (NLP) hanno messo in luce la necessità di individuare le frasi degne di essere riassunte nel testo sorgente, per affrontare il problema del trattamento di grandi quantità di informazioni. Intuitivamente, ogni frase può essere etichettata come rilevante o meno per il riassunto finale, al fine di addestrare un modello a riconoscere tali frasi salienti. Tuttavia, a causa della mancanza di un'etichetta di rilevanza veritiera, è necessario marcare euristicamente la rilevanza delle frasi in input rispetto al riassunto di riferimento, attuando una strategia di etichettatura morbida (soft labeling).

Algorithm 1 Soft Labeling

Input:
$X = \{x_1, \ldots, x_x\}$ ▷ Input sentences
$Y = \{y_1, \ldots, y_y\}$ ▷ Output sentences
Parameters: M ▷ Similarity metric
Output: S ▷ Set of scores

```
S ← ∅
for x_i ∈ X do
    s ← ∅
    for y_i ∈ Y do
        s.append(M(x_i, y_i))
    end for
    S.append(max(s))
end for
return S
```

Concretamente, siano $X = \{x_1, \ldots, x_x\}$ e $Y = \{y_1, \ldots, y_y\}$ l'input lungo e il riassunto corrispondente, rispettivamente, dove ogni $x_i \in X$ e $y_i \in Y$ è una frase. Per ogni x_i, generiamo un punteggio di rilevanza $\in [0, 1]$ calcolando la similarità rispetto a ciascuna y_i (e poi prendendo il miglior valore), utilizzando un algoritmo greedy (Algoritmo 1). Abbiamo testato diverse metriche di valutazione (ad esempio, BLEU e ROUGE) come funzioni di similarità su MULTI-LEXSUM (SHORT), utilizzando 100 campioni per i set di training e validazione. Abbiamo impiegato PRIMERA come modello di base per la summarization, un trasformatore con complessità lineare nella lunghezza dell'input pre-addestrato con un obiettivo specifico per il Multi-Document Summarization (MDS). Tecnicamente, dopo aver assegnato un punteggio di rilevanza del riassunto a ciascuna $x_i \in X$, diamo a PRIMERA solo le frasi (nell'ordine di comparsa nella sorgente) con i punteggi più alti fino alla dimensione massima di input del modello (ovvero, 4096 token). La Figura 3.5 indica che la migliore

metrica per l'etichettatura morbida è ROUGE-2 F1, quindi è stata utilizziamo tale metrica per etichettare la rilevanza delle frasi. Notiamo che la Figura 3.5 riporta i risultati ottenuti.

Metric	**R-1**$_{f1}$	**R-2**$_{f1}$	**R-L**$_{f1}$
BLEU	43.62	19.18	28.58
R1-F1	44.06	19.33	29.32
R1-P	40.97	16.96	26.51
R2-F1	**45.29**	**20.20**	**30.19**
R2-P	43.32	19.20	29.14
RL-F1	43.18	19.34	28.30
RL-P	43.90	19.14	29.15

Figura 3.5: Risultati del Multi-Document Summarization (MDS) su MULTI-LEXSUM (SHORT). Vengono forniti a PRIMERA input diversi ottenuti utilizzando diverse metriche come funzioni di similarità per l'etichettatura morbida (soft labeling). [1]

3.2.2 Da Documento a Grafo

Il modulo document-to-graph ha l'obiettivo di creare un grafo eterogeneo semantico attraverso i seguenti passaggi (vedere Figura 2):

1. **Estrazione delle parole chiave:** Inizialmente, rimuoviamo le stopwords in inglese e i termini specifici del dominio generale che compaiono in più del 40% dell'input (in ogni documento per MDS). Successivamente, utilizziamo KEYBERT, un metodo leggero rispetto ad altre soluzioni più esigenti in termini di risorse, per selezionare fino a k parole chiave. Per MDS, estraiamo k parole chiave per ogni documento nel cluster e combiniamo queste liste di parole chiave in un set unico, eliminando i duplicati.

2. **Filtraggio delle frasi:** Suddividiamo l'input in frasi e selezioniamo quelle con almeno una parola chiave. Inoltre, scegliamo n frasi che compaiono prima e dopo quella selezionata come contesto. Per MDS, definiamo il cluster di z fonti. Selezioniamo quindi il contesto di x_b e concateniamo le frasi.

3. **Incorporamento di frasi e parole chiave:** Produciamo l'incorporamento di tutte le parole chiave e le frasi chiave utilizzando DISTILROBERTA, un PLM caratterizzato da pochi parametri che rende la nostra soluzione efficiente in termini di memoria GPU e occupazione. Questo modello è già pre-addestrato per creare incorporamenti di frasi utilizzando un obiettivo di apprendimento auto-supervisionato contrastivo. Generiamo il vettore finale e_{x_i} mediando gli incorporamenti dei token.

4. **Creazione del grafo:** Tutti gli incorporamenti di parole chiave (KE) e frasi (SE) diventano i nodi del nostro grafo. Ispirati da precedenti lavori, utilizziamo KE come super-nodi, il che significa che tutte le frasi contenenti una parola chiave hanno un arco bidirezionale Keyword Edge con il nodo della parola chiave. Poi, aggiungiamo archi bidirezionali Positional Edges tra due SE se appaiono consecutivamente nella Fonte Infine, aggiungiamo Semantic Edges tra e_i ed e_j se la loro similarità coseno è superiore a una soglia t. Il grafo creato ha tanti nodi quanti sono il numero di frasi e parole chiave.

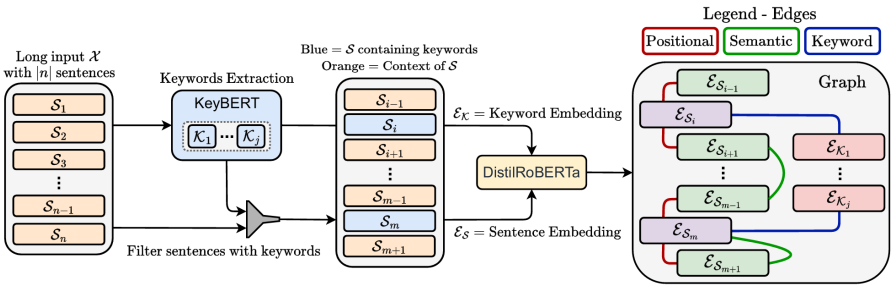

Figura 3.6: La pipeline adottata da G-SEEK per produrre un grafo eterogeneo semantico a partire da un input lungo. Il documento viene fornito a KEYBERT che crea un insieme di parole chiave uniche. Quindi, (i) le frasi contenenti almeno una parola chiave, (ii) il loro contesto (frasi immediatamente successive o precedenti), (iii) e le parole chiave vengono trasformate in incorporamenti utilizzando DISTILROBERTA, diventando i nuovi nodi del grafo collegati con diversi archi significativi. Fonte [1]

3.2.3 Apprendimento dei Punteggi di Rilevanza

Si utilizza un Graph Attention Network (GAT) per apprendere le relazioni tra i nodi nel grafo (informazioni strutturali) insieme alle informazioni nei loro nodi (informazioni semantiche). Considerando gli archi che collegano i nodi, un GAT può imparare a propagare le informazioni attraverso le frasi, comprendendo meglio il contesto e il significato di ciascuna frase. Inoltre, i GAT possono gestire in modo efficiente grafi enormi e complessi. In questo studio, si è utilizzato il GAT per identificare le frasi più critiche assegnando un punteggio di rilevanza positivo illimitato a ciascun nodo. Tecnicamente, questo modulo ha i seguenti strati:

- **Strato di Riproiezione:** Consiste di due strati feed-forward lineari (FFL) che apprendono come riproiettare (x′) le embedding dei nodi x

nello spazio vettoriale. Meccanicamente, espande la dimensione dx delle n embedding in input ($R^{n\times768}$) di un fattore denominato Boom Factor (BF), ispirato all'architettura del trasformatore:

$$x' = FFL\sigma(FFL\gamma(x, dx \cdot BF), dx) \tag{3.1}$$

dove σ e γ sono parametri apprendibili di diversi strati lineari.

- **Strato GAT:** Sfrutta le informazioni strutturali per arricchire la semantica dei nodi. L'output x_b è generato interpolando tutte le righe di x′ pesate da un punteggio a ($R^{n\times768}$):

$$x_{b_{i,j}} = a(Wx'_i, Wx'_j) \tag{3.2}$$

- **Strato di Punteggio:** Comprende due strati lineari per ridurre la dimensione di ciascun nodo (x_b) a un unico numero reale (s) utilizzato come punteggio di rilevanza della frase associata al nodo (R^n):

$$s = FFL\beta(FFL\theta(x, d_b x_b \cdot BF), 1) \tag{3.3}$$

Per supervisionare il modello GAT, si sfruttano le soft labels (l) spiegate nella Sezione 3.2.1. Tecnicamente, si addestra il modello a produrre punteggi (s) che minimizzano la seguente perdita:

$$L_{mse} = (s - l)^2 \tag{3.4}$$

3.2.4 Summarization Pipeline

Una volta trasformato l'input in un grafo e dopo che il modulo GAT assegna un punteggio a ciascuna frase, estraiamo quelle più rilevanti e utilizziamo un modello di linguaggio pre-addestrato (PLM) generativo per generare il riassunto finale. Tecnicamente, in base ai loro punteggi di rilevanza, selezioniamo le frasi più salienti e creiamo un testo in input per il modello che contiene meno token rispetto alla sua dimensione massima di input. Di conseguenza, il nuovo input comprende frasi nell'ordine di comparsa nella fonte testuale originale.

Addestriamo il modello di summarization utilizzando la perdita standard di cross-entropia, che richiede al modello di predire il prossimo token w_i del target Y dato X e i precedenti token target $w_{1:i-1}$, come segue:

$$L_{ce} = -\sum_{i=1}^{|Y|} \log p_\tau(w_i | w_{1:i-1}, X) \tag{3.5}$$

dove τ indica i parametri del modello e p è la probabilità predetta sul vocabolario.

3.2.5 Procedura di Addestramento

Il processo di addestramento in G-SEEK inizia con l'inizializzazione del tokenizer e del modello, impostato per elaborare input 4 a 32 bit con la precisione mista e configurato con la precisione desiderata su un dispositivo specifico.

Si procede quindi con la creazione di un *Dataloader*, che si occupa di organizzare il dataset di addestramento in batch conformi alle necessità del modulo GNN. Sono definiti l'ottimizzatore e il learning rate scheduler per guidare l'ottimizzazione del modello, mentre viene configurato separatamente l'ottimizzatore per il modulo GNN, dimostrando la natura composita di questa procedura di addestramento.

Algorithm 2 Training Procedure

Input: train_dataset, eval_dataset, gnn_module, max_source_length, model_id, batch_size, lr, device, gnn_lr, gnn_optim, gnn_loss
Output: trained model

```
tokenizer ← Tokenizer(model_id)
model ← Model(model_id, device)
dataloader ← DataLoader(train_dataset, batch_size, collate_fn_for_gnn)
optimizer ← Adam(model.parameters, lr)
lr_scheduler ← get_linear_schedule(optimizer)
for each epoch do
    model.train()
    for each batch in train_dataloader do
        input_ids, attention_mask ← get_salient_content_using_gnn(
            batch[document],tokenizer,gnn_module,gnn_loss,gnn_optim)
        labels ← tokenizer(summary,max_source_length)["input_ids"]
        outputs ← model(input_ids, attention_mask, labels)
        outputs.loss.backward()
        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
    end for
    test(model_id, tokenizer, eval_dataset, document,
         summary, epoch, gnn_module)
end for
return model
```

All'interno di ogni epoca, il modello è impostato in modalità di addestramento. Per ogni batch del *Dataloader*, i testi di input e di riassunto vengono

estratti e pre-elaborati. La GNN viene utilizzata per determinare i contenuti salienti che, insieme alle maschere di attenzione corrispondenti, vengono forniti al modello.

La fase centrale dell'addestramento è rappresentata dal passaggio in avanti (*feed-forward*), in cui vengono calcolati gli output e il loss del modello, seguito dal passaggio all'indietro (*feed-backward*) per il calcolo dei gradienti. Segue un passo di ottimizzazione in cui l'ottimizzatore aggiorna i pesi del modello. Il learning rate scheduler aggiusta il tasso di apprendimento e, infine, i gradienti vengono azzerati.

Dopo ogni epoca, il modello viene valutato su un set di validazione per monitorarne le prestazioni. Questo ciclo viene ripetuto per un numero prestabilito di epoche, con l'obiettivo di affinare progressivamente la capacità del modello di generare riassunti accurati e pertinenti.

3.3 Analisi dei Dataset

I dataset svolgono un ruolo cruciale nella ricerca e sviluppo nel campo del Natural Language Processing. Essi forniscono il contesto e i dati necessari per addestrare, testare e validare i modelli. In questo studio, sono stati utilizzati diversi dataset per valutare le performance di vari modelli di summarizzazione, ciascuno con le sue caratteristiche uniche e sfide specifiche. Di seguito, vengono esaminate le peculiarità di alcuni dei principali dataset utilizzati, evidenziando i loro aspetti distintivi e la loro rilevanza per questo studio.

3.3.1 Multi-LexSum

Il dataset Multi-LexSum è una collezione unica composta da 9.280 riassunti di casi legali. La sua particolarità risiede nella diversità dei riassunti proposti: mentre alcuni sono estremamente concisi, limitandosi a una sola frase, altri si estendono attraverso narrazioni dettagliate di vari paragrafi, spesso superando le cinquecento parole. Questa varietà rappresenta una sfida di Summarization multi-documento, considerando anche la lunghezza considerevole dei documenti sorgente, che in alcuni casi superano le duecento pagine. Un altro aspetto distintivo di Multi-LexSum è la sua autenticità e accuratezza. A differenza di molti altri dataset, tutti i riassunti sono stati redatti da professionisti del settore legale, tra cui avvocati e studenti di giurisprudenza, seguendo linee guida rigorose e sottoposti a revisioni da parte di esperti per garantire la massima qualità.

Struttura del dataset

sources: Colonna che rappresenta la sequenza di documenti sorgente da cui è stato estratto il riassunto. Potrebbe contenere riferimenti, citazioni o link ai documenti originali, fornendo un contesto dettagliato per ogni riassunto.

summary/long: Colonna che contiene i riassunti più lunghi (circa 650 parole) e dettagliati dei casi legali, spesso estesi a diversi paragrafi. Questi riassunti forniscono una panoramica completa e dettagliata del caso.

summary/short: Colonna che contiene versioni più concise dei riassunti (circa 150 parole), riducendo le informazioni a poche frasi o paragrafi. Sono utili per ottenere una comprensione rapida del caso senza entrare in dettagli eccessivi.

summary/tiny: Colonna dove si trovano i riassunti "estremi" (circa 25 parole), che condensano l'essenza del caso in una sola frase o in poche frasi. Questi riassunti sono utili per avere una visione immediata e sintetica del caso in esame.

3.3.2 GovReport

GovReport è un dataset destinato alla Summarization di documenti lunghi, caratterizzato da documenti e riassunti significativamente più estesi. È composto da rapporti redatti da agenzie di ricerca governative, tra cui il Congressional Research Service e l'U.S. Government Accountability Office (Ufficio di Rendicontazione del Governo degli Stati Uniti).

Struttura del dataset

- **report:** Una stringa che contiene il testo completo del rapporto, suddiviso in varie sezioni, ciascuna con il proprio titolo e paragrafi.
- **summary:** Una stringa che presenta un riassunto conciso del rapporto, preservando le informazioni essenziali e le conclusioni chiave.

3.3.3 Multi-News

Multi-News è un dataset che comprende articoli di notizie e riassunti scritti da umani relativi a questi articoli, provenienti dal sito newser.com. Ogni riassunto è stato redatto professionalmente da editori e include collegamenti agli articoli originali citati. Questa particolarità fornisce un ulteriore livello

di contestualizzazione e verifica, consentendo agli utenti di esplorare le fonti originali e confrontarle con i riassunti forniti.

La scelta di includere articoli da un sito di notizie come newser.com rappresenta un tentativo di catturare la natura dinamica e multifaceted delle notizie contemporanee. Gli articoli di notizie tendono ad essere più brevi e diretti rispetto ai documenti legali o ai rapporti governativi, ma presentano sfide uniche nella Summarization a causa della loro urgenza, rilevanza e varietà di stili e toni.

Struttura del dataset

document: Questa colonna contiene il testo degli articoli di notizie. Gli articoli sono separati da un token speciale "|||||", che indica la transizione da un articolo all'altro. Questo formato consente di avere una visione combinata di più articoli correlati, offrendo una panoramica completa di un evento o argomento di notizie.

summary: Questa colonna contiene il riassunto delle notizie. I riassunti offrono una sintesi chiara e concisa degli articoli originali, permettendo ai lettori di comprendere rapidamente i punti salienti e le questioni chiave senza dover leggere l'intero set di articoli.

3.3.4 WikiCatSum

WikiCatSum è un dataset di Summarization in inglese che si estende su tre domini: animali, aziende e film. Fornisce più paragrafi di testo abbinati a un riassunto dei paragrafi. Questa struttura multi-paragrafo offre una sfida unica per la Summarization, in quanto richiede una comprensione approfondita e la capacità di condensare informazioni da diverse sezioni di testo in un riassunto coerente.

La scelta di concentrarsi su tre domini specifici permette di esplorare le sfide e le peculiarità della Summarization in contesti diversi. Ad esempio, la Summarization di articoli sugli animali potrebbe richiedere una focalizzazione sulle caratteristiche biologiche e sul comportamento, mentre gli articoli sulle aziende potrebbero concentrarsi su dettagli finanziari, storici e strategici.

Struttura del dataset

title: Corrisponde al titolo dell'articolo di Wikipedia da cui sono stati estratti i paragrafi e il riassunto. Il titolo fornisce un contesto iniziale e indica l'argomento principale dell'articolo.

paragraphs: Questa colonna contiene una lista ordinata di paragrafi provenienti dall'insieme di testi estratti. Questi paragrafi rappresentano le diverse sezioni o aspetti dell'articolo originale di Wikipedia.

summary: Questa colonna è costituita da una lista di frasi insieme alla loro corrispondente etichetta di argomento. Queste etichette di argomento forniscono ulteriori dettagli sul contenuto e l'importanza di ciascuna frase nel riassunto, aiutando a comprendere la struttura e la gerarchia delle informazioni.

3.3.5 WCEP

Il dataset WCEP per la Summarization multi-documento (MDS) consiste in riassunti brevi, scritti da persone, riguardanti eventi di attualità, ottenuti dal Wikipedia Current Events Portal (WCEP). Ogni riassunto è associato a un cluster di articoli di notizie correlati a un evento. Questi articoli comprendono fonti citate dagli editor su WCEP e sono arricchiti con articoli ottenuti automaticamente dal dataset Common Crawl News.

La particolarità di WCEP risiede nella sua natura multi-documento. Invece di basarsi su un singolo articolo come sorgente di informazioni, il dataset si concentra sull'aggregazione di contenuti da vari articoli per fornire un riassunto completo e bilanciato degli eventi correnti. Questo approccio riflette le sfide reali della Summarization nell'era dell'informazione, dove è comune attingere da molteplici fonti per ottenere una visione completa di un evento o di una notizia.

Struttura del dataset

document: Questa colonna contiene una stringa o una lista che rappresenta il corpo di un insieme di documenti. Questi documenti rappresentano le diverse fonti di informazione che sono state utilizzate per creare il riassunto finale.

summary: Questa colonna contiene una stringa che rappresenta il riassunto astratto dell'insieme di documenti. Questi riassunti sono scritti da editori umani e riflettono una sintesi accurata e bilanciata delle informazioni presenti nei documenti sorgente.

3.4 Analisi delle GNN utilizzate

Nell'ambito del Natural Language Processing, le Graph Neural Networks (GNNs) stanno emergendo come potenti strumenti per catturare e rappresentare

relazioni complesse tra entità. G-SEEK sfrutta diverse varianti di GNNs per modellare e analizzare le relazioni tra parole chiave, frasi e contesto nei documenti. Di seguito, presentiamo una panoramica delle GNNs utilizzate:

3.4.1 Graph Attention Network (GAT)

Le Graph Attention Networks (GAT) sono state proposte come una soluzione per incorporare meccanismi di attenzione nelle Graph Neural Networks (GNN). Questi meccanismi di attenzione permettono ai modelli di pesare diversamente i vicini di un nodo, consentendo una migliore aggregazione delle informazioni.

Caratteristiche principali

La caratteristica distintiva delle GAT è l'uso di meccanismi di attenzione per pesare i contributi dei nodi vicini durante l'aggregazione. Questa attenzione è basata sulla relazione tra i nodi e non è fissa, ma è appresa durante l'addestramento. Ecco alcune delle principali caratteristiche delle GAT:

- **Attenzione basata sulle caratteristiche:** Le GAT utilizzano le caratteristiche dei nodi per calcolare i pesi di attenzione. Questo significa che nodi simili in termini di caratteristiche avranno pesi di attenzione simili.
- **Multi-head Attention:** Simile alla Transformer architecture, le GAT utilizzano multi-head attention per catturare diversi tipi di relazioni tra i nodi.
- **Flessibilità:** Le GAT possono essere facilmente integrate in altre architetture GNN.

Formulazione Matematica

Dato un grafo $G = (V, E)$ e un nodo v, l'obiettivo delle GAT è di aggregare le informazioni dai vicini di v in modo ponderato:

$$e_{uv} = \text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{H}_u \parallel \mathbf{W}\mathbf{H}_v]\right)$$

dove $\mathbf{H}_u$ e $\mathbf{H}_v$ sono le rappresentazioni dei nodi u e v, e $\mathbf{W}$ e $\mathbf{a}$ sono matrici di pesi.

I pesi di attenzione sono poi normalizzati:

$$\alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{k \in \mathcal{N}(v)} \exp(e_{uk})}$$

Infine, la nuova rappresentazione del nodo v viene calcolata come:

$$\mathbf{H}'_v = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv} \mathbf{W}\mathbf{H}_u\right)$$

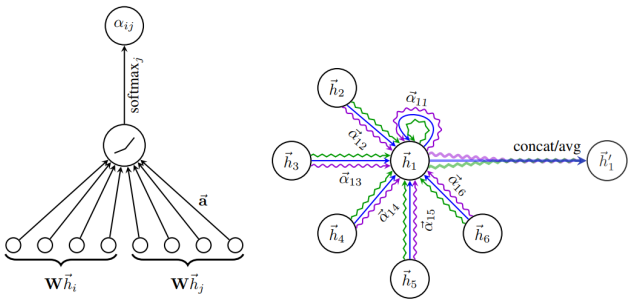

Figura 3.7: Sinistra: Il meccanismo di attenzione $a(W\tilde{hi}, W\tilde{hj})$ impiegato dal nostro modello, parametrizzato da un vettore di peso $\tilde{a} \in R^{2F_0}$, applicando una funzione di attivazione LeakyReLU. Destra: Un'illustrazione dell'attenzione multi-testa (con $K = 3$ teste) effettuata dal nodo 1 sul suo vicinato. Stili e colori delle frecce diversi denotano calcoli di attenzione indipendenti. Le caratteristiche aggregate da ogni testa sono concatenate o mediate per ottenere $\tilde{H}_1^0$. Fonte [25]

Importanza delle GAT in G-SEEK per la Summarization di Testi

Le Graph Attention Networks (GAT) svolgono un ruolo cruciale in G-SEEK quando si tratta di Summarization di testi. La capacità di GAT di pesare differentemente i vicini di un nodo durante l'aggregazione delle informazioni, basandosi su meccanismi di attenzione, rende queste reti estremamente efficaci nella cattura delle sottigliezze e delle relazioni semantiche all'interno dei testi.

La Summarization richiede che un modello identifichi e dia priorità a porzioni di testo basate sulla loro rilevanza e importanza nel contesto complessivo. Le GAT, con i loro meccanismi di attenzione, sono naturalmente predisposte per questo tipo di task. I pesi di attenzione appresi dal modello possono indicare quali parti del testo (o quali frasi) sono considerate più importanti o rilevanti durante la Summarization. Questa ponderazione dinamica delle informazioni consente a G-SEEK di produrre riassunti che catturano efficacemente i punti salienti e le informazioni chiave del testo originale.

Inoltre, poiché la Summarization di testi spesso coinvolge documenti con molte frasi interconnesse e informazioni correlate, la capacità delle GAT di considerare e pesare le relazioni tra diverse parti del testo aiuta a costruire rappresentazioni più coerenti e complete. Questo, a sua volta, conduce a

riassunti che sono sia informativi che leggibili, garantendo che le informazioni essenziali vengano catturate mentre si evitano ripetizioni o informazioni non essenziali.

3.4.2 Graph Convolutional Network (GCN)

Le GCN sono una delle prime e più semplici varianti di GNN. Aggregano informazioni dai nodi vicini attraverso operazioni di convoluzione. Anche se semplici, le GCN possono catturare efficacemente le relazioni locali nel grafo. A differenza dei metodi tradizionali di apprendimento su grafi che dipendono pesantemente dalla costruzione manuale delle caratteristiche, le GCN apprendono in modo end-to-end le rappresentazioni dei nodi attraverso la convoluzione nei grafi. La loro architettura si basa su un'operazione di aggregazione locale dei nodi vicini, seguita da una trasformazione non lineare. Questo processo viene iterato per diversi livelli, permettendo alle GCN di catturare le informazioni dai vicini a diverse distanze. Nel dettaglio, l'operazione fondamentale delle GCN può essere descritta come:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

Nell'equazione:

- $H^{(l)}$: stato nascosto (o attributi del nodo quando $l = 0$).
- $\tilde{D}$: matrice dei gradi.
- $\tilde{A}$: matrice di adiacenza (con auto-connessioni).
- $W^{(l)}$: pesi addestrabili al livello l.
- σ: funzione di attivazione, ad esempio ReLU.
- l: numero del livello.

Le GCN sono state dimostrate essere particolarmente efficaci in compiti di classificazione di nodi, dove l'obiettivo è predire una label per ogni nodo in un grafo, sfruttando sia le sue caratteristiche che la sua posizione nel grafo.

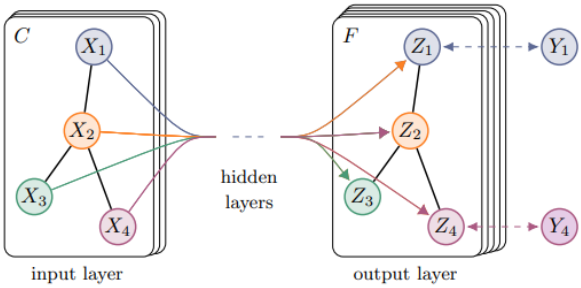

Figura 3.8: Rappresentazione schematica di una Graph Convolutional Network (GCN) multi-strato per l'apprendimento semi-supervisionato con C canali in ingresso e F mappe di caratteristiche nello strato di uscita. La struttura del grafo (bordi mostrati come linee nere) è condivisa tra gli strati, le etichette sono denotate da Y_i. Fonte [26]

Nella figura, possiamo osservare un grafo con nodi collegati da linee nere, rappresentanti gli edge del grafo. Questa struttura rappresenta il dataset su cui opera la GCN.

La GCN è composta da più layer, attraverso i quali ogni nodo nel grafo passa. Ad ogni attraversamento di un layer, l'informazione di un nodo viene aggiornata aggregando le informazioni dai suoi nodi vicini e da se stesso.

- **Input**: Inizialmente, ogni nodo possiede C canali di input. Tali canali potrebbero rappresentare diverse caratteristiche o proprietà associati al nodo.

- **Layers intermedi**: Con l'avanzamento attraverso la GCN, le informazioni dei nodi vengono aggregate e trasformate. In ogni layer, le informazioni di un nodo vengono fuse con quelle dei suoi vicini utilizzando pesi appresi e funzioni di attivazione.

- **Output**: Alla conclusione della GCN, ogni nodo produce F mappe di caratteristiche, che rappresentano l'output della rete per quel nodo in particolare.

Oltre alla struttura del grafo, alcune etichette Y_i potrebbero essere associate a determinati nodi. Queste etichette rappresentano informazioni note riguardo alcuni nodi e vengono utilizzate nell'apprendimento semi-supervisionato. In tale contesto, solo un sottoinsieme di nodi ha etichette conosciute, e l'obiettivo è quello di utilizzare queste etichette note per inferire informazioni sui nodi non etichettati.

Classificazione Semi-Supervisionata dei Nodi

Nel contesto delle Graph Convolutional Networks (GCN), una delle applicazioni fondamentali è la classificazione semi-supervisionata dei nodi. Questo approccio combina le informazioni provenienti da nodi etichettati e non etichettati per migliorare le performance della classificazione su grafi.

Quando si parla di "nodi" in un grafo che rappresenta un documento, ogni nodo può rappresentare una frase o un segmento del documento. Le relazioni o gli archi tra i nodi possono rappresentare connessioni semantiche, sequenze o altre relazioni rilevanti tra le frasi.

Il vantaggio principale della classificazione semi-supervisionata con GCN è che non si basa esclusivamente sui nodi etichettati per l'addestramento. Anche se solo una piccola parte dei nodi in un grafo è etichettata, le GCN possono sfruttare la struttura del grafo e le informazioni dai nodi non etichettati per migliorare la classificazione.

Ecco come funziona:

1. **Propagazione delle informazioni**: Utilizzando la struttura del grafo e le caratteristiche dei nodi, le GCN propagano le informazioni attraverso i nodi durante l'addestramento. Ciò consente di aggregare informazioni dai vicini di ogni nodo, arricchendo così la rappresentazione di ogni nodo con informazioni dal suo contesto.

2. **Utilizzo di nodi etichettati e non etichettati**: Durante l'addestramento, le GCN utilizzano sia i nodi etichettati che quelli non etichettati. I nodi etichettati forniscono un segnale di addestramento diretto, mentre i nodi non etichettati aiutano a catturare e propagare le informazioni contestuali attraverso il grafo.

3. **Predizione**: Una volta addestrata, la GCN può essere utilizzata per prevedere le etichette dei nodi non etichettati. Questa previsione si basa sia sulle caratteristiche del nodo che sulle informazioni aggregate dai suoi vicini.

Nel contesto della Summarization del testo, la classificazione semi-supervisionata dei nodi potrebbe essere utilizzata per determinare la rilevanza delle frasi in un documento. Ad esempio, le frasi potrebbero essere classificate come rilevanti o non rilevanti per un riassunto, utilizzando un piccolo set di frasi etichettate come training e sfruttando le relazioni tra le frasi per migliorare le prestazioni della classificazione su tutto il documento.

Importanza delle GCN in G-SEEK per la Summarization di Testi

La Graph Convolutional Network (GCN) si dimostra di particolare efficacia nel contesto di G-SEEK per la Summarization di testi. Di seguito sono riportate le ragioni chiave della sua rilevanza:

1. **Rappresentazione delle Relazioni Locali:** Le GCN sono particolarmente abili nell'apprendere dalla struttura locale di un grafo. Questa caratteristica risulta fondamentale in G-SEEK, dove la capacità di identificare le relazioni tra frasi vicine è cruciale per comprenderne la rilevanza all'interno di un documento.

2. **Apprendimento Diretto dalle Caratteristiche dei Nodi:** Le GCN utilizzano le informazioni testuali intrinseche delle frasi, consentendo di combinare il contenuto e la struttura del grafo per valutare l'importanza di una determinata frase.

3. **Analisi Gerarchica del Documento:** La capacità delle GCN di catturare informazioni da nodi a diverse distanze consente di comprendere la rilevanza di una frase considerando sia il suo contesto diretto sia le sue relazioni con l'intero testo.

4. **Integrazione con Modelli di Linguaggio:** Le rappresentazioni ottenute dalla GCN possono essere facilmente integrate con potenti modelli di linguaggio per generare riassunti accurati, sfruttando una profonda comprensione delle relazioni tra le frasi.

3.4.3 Graph Isomorphism Network (GIN)

Le Graph Isomorphism Networks (GINs) rappresentano un avanzamento significativo nel campo delle Graph Neural Networks (GNNs) in quanto sono state progettate per catturare efficacemente sottografi isomorfi, garantendo che due sottografi strutturalmente identici (isomorfi) abbiano la stessa rappresentazione nell'embedding. Questa capacità di discernere sottili differenze strutturali rende le GINs strumenti potenti per modellare dati complessi rappresentati come grafi.

Caratteristiche Principali:

- **Rappresentazione Unica per Sottografi Isomorfi:** La GIN garantisce che sottografi isomorfi abbiano la stessa rappresentazione, risolvendo uno dei principali problemi nelle GNNs tradizionali.

- **Aggregazione Ponderata:** Al centro della GIN c'è un'operazione di aggiornamento ponderata. Per un nodo v, la sua nuova rappresentazione viene calcolata come:

$$h_v^{(k)} = \text{MLP}^{(k)} \left((1 + \epsilon) h_v^{(k-1)} + \sum_{u \in \text{N}(v)} h_u^{(k-1)} \right)$$

  dove $h_v^{(k)}$ è la rappresentazione del nodo v al livello k, $\text{N}(v)$ rappresenta l'insieme dei nodi vicini di v, e ϵ è un parametro di ponderazione.

- **Espressività e Cattura di Relazioni Semantiche Sottili:** La formula di aggiornamento, insieme all'introduzione del parametro di ponderazione ϵ, permette alla GIN di catturare relazioni semantiche sottili tra nodi e sottografi, rendendola particolarmente efficace per problemi che richiedono una comprensione dettagliata della struttura del grafo.

- **Fondamenti Teorici Solidi:** La progettazione delle GIN è sostenuta da robusti principi teorici, concentrati in particolare sul problema dell'isomorfismo del grafo. Questo assicura che la rete possa effettivamente distinguere tra grafi non isomorfi, indipendentemente dalla loro somiglianza superficiale.

Importanza delle GIN in G-SEEK per la Summarization di Testi

Le Graph Isomorphism Networks (GIN) sono particolarmente rilevanti nel contesto di G-SEEK per la Summarization di testi. La capacità delle GIN di catturare relazioni strutturali e attributi nodali nei grafi le rende particolarmente potenti per analizzare la struttura complessa dei documenti testuali.

Nel contesto della Summarization, ogni documento può essere rappresentato come un grafo, dove ogni nodo corrisponde a una frase o a un'entità e gli archi rappresentano le relazioni tra di essi. G-SEEK, utilizzando GIN, è in grado di catturare sottili relazioni semantiche tra diverse parti del testo, permettendo una rappresentazione più ricca e dettagliata del documento.

I vantaggi principali nell'utilizzare GIN in G-SEEK includono:

- **Rappresentazione Dettagliata**: Le GIN catturano con precisione le relazioni intrinseche tra le frasi, offrendo una rappresentazione dettagliata e profonda dei documenti.

- **Invarianza alle Permutazioni**: Una delle caratteristiche chiave delle GIN è la loro capacità di essere invarianti alle permutazioni dei nodi,

garantendo che la rappresentazione del documento non sia influenzata dall'ordine delle frasi.

- **Adattabilità**: Le GIN sono in grado di adattarsi a diverse topologie di grafo, rendendole adatte a documenti di varia struttura e lunghezza.

3.4.4 DeepGCN

Le Graph Convolutional Networks (GCN) hanno dimostrato un notevole successo nel campo dell'apprendimento profondo su grafi. Tuttavia, l'espansione delle GCN a modelli profondi può portare a problemi come la scomparsa o l'esplosione del gradiente, limitando così la profondità delle GCN che possono essere efficacemente addestrate. DeepGCN affronta questo problema introducendo tecniche che consentono di addestrare modelli GCN più profondi. Ecco alcune delle tecniche:

- **Connessioni Residue:** Una delle principali innovazioni delle DeepGCN è l'introduzione delle connessioni residue, ispirate dalla popolare architettura di rete neurale ResNet utilizzata per le immagini. Queste connessioni permettono di bypassare uno o più strati, garantendo un flusso di informazioni ininterrotto attraverso la rete e aiutando a prevenire problemi come la scomparsa del gradiente. Inoltre, le connessioni residue possono ridurre la quantità di memoria necessaria poiché condividono i pesi tra gli strati.

- **Aggregazione Adattiva:** L'aggregazione adattiva in DeepGCN consente al modello di concentrarsi su informazioni rilevanti, riducendo la necessità di elaborare dati irrilevanti o ridondanti. Questo può portare a una riduzione dei requisiti di memoria poiché solo le informazioni essenziali vengono propagate attraverso la rete.

- **Normalizzazione dei Nodi:** La normalizzazione dei nodi in DeepGCN aiuta a stabilizzare i valori all'interno della rete, prevenendo l'esplosione o la scomparsa del gradiente. Questa stabilizzazione può ridurre la necessità di memorizzare valori estremamente grandi o piccoli, contribuendo a ridurre i requisiti di memoria.

Importanza delle DeepGCN in G-SEEK per la Summarization di Testi

La Summarization di testi richiede una comprensione profonda e dettagliata del contenuto di un documento. Per catturare efficacemente le sfumature e

le relazioni intrinseche tra le frasi e le parole, è essenziale avere una rappresentazione che possa penetrare attraverso le molteplici dimensioni semantiche e strutturali del testo. Le DeepGCN, con la loro architettura profonda e le tecniche innovative, offrono un mezzo potente per realizzare questo compito.

Le caratteristiche principali delle DeepGCN, come le connessioni residue e l'aggregazione adattiva, le rendono particolarmente adatte per la Summarization di testi. Questi meccanismi garantiscono che le informazioni fluiscono efficacemente attraverso i vari strati della rete, permettendo al modello di catturare e rappresentare relazioni complesse all'interno del documento.

Inoltre, la capacità delle DeepGCN di avere molteplici strati senza un aumento significativo dei requisiti di memoria consente di modellare relazioni a più livelli all'interno del testo. Questo è fondamentale per la Summarization, dove le relazioni tra le frasi possono esistere a diversi livelli di granularità, dalla coerenza di alto livello alle relazioni semantiche sottili tra parole o frasi adiacenti.

L'approccio adottato da DeepGCN per gestire grafici di grandi dimensioni consente inoltre di trattare documenti estesi, garantendo che ogni parte del testo sia considerata nella generazione della Summarization. Questa capacità di scalare garantisce che la Summarization sia completa e rappresentativa del contenuto originale.

3.4.5 GraphSAGE

GraphSAGE (Graph Sample and Aggregation) rappresenta un'innovazione significativa nell'ambito delle Graph Neural Networks (GNN). Mentre le GNN tradizionali si concentrano sull'apprendimento di rappresentazioni per grafi interi o per sottografi noti, GraphSAGE è stato progettato per generare rappresentazioni per nodi in grafi di grandi dimensioni, incorporando informazioni dai loro vicini.

- **Aggregazione e Campionamento:** La caratteristica chiave di GraphSAGE è la sua capacità di campionare e aggregare informazioni dai vicini di un nodo. Invece di basarsi su matrici di adiacenza statiche, GraphSAGE campiona un numero fisso di vicini per nodo e aggrega le loro caratteristiche per generare una rappresentazione.

- **Flessibilità nell'Aggregazione:** GraphSAGE supporta diverse funzioni di aggregazione, tra cui media, LSTM e pooling, offrendo una vasta gamma di opzioni per incorporare le informazioni dei vicini in base alla natura del grafo e del compito specifico.

- **Rappresentazioni Induttive:** A differenza delle GNN che richiedono l'intero grafo per generare rappresentazioni, GraphSAGE apprende funzioni di aggregazione in modo che possa generare rappresentazioni per nodi non visti durante la fase di addestramento. Ciò lo rende particolarmente utile per grafi dinamici che crescono nel tempo.

- **Scalabilità:** La capacità di campionare vicini rende GraphSAGE altamente scalabile, consentendo l'addestramento su grafi di grandi dimensioni. Questa scalabilità si estende anche a grafi con milioni di nodi e spigoli.

- **Applicazioni e Ricerca:** GraphSAGE ha dimostrato efficacia in una serie di compiti di apprendimento su grafi, tra cui classificazione di nodi, previsione di legami e altri compiti correlati ai grafi. Un esempio di successo è rappresentato da PinSage, una variante di GraphSAGE, utilizzata da Pinterest per la raccomandazione di contenuti.

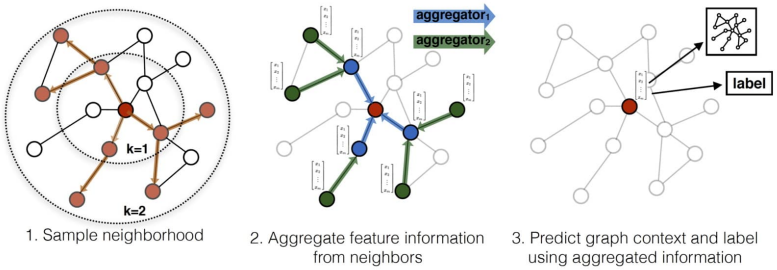

Figura 3.9: Tecnica di aggregazione per il passaggio di messaggi. Fonte [15]

Importanza delle GraphSAGE in G-SEEK per la Summarization di Testi

Nel contesto della Summarization di testi, l'importanza delle relazioni tra frasi e parole è fondamentale per comprendere il contenuto e l'essenza di un documento. Le Graph Neural Networks, come GraphSAGE, giocano un ruolo cruciale in questo scenario, offrendo un meccanismo efficace per catturare e rappresentare tali relazioni.

La capacità di GraphSAGE di campionare e aggregare informazioni dai vicini di un nodo la rende particolarmente adatta per analizzare le relazioni tra frasi in un testo. Quando si considera ogni frase o parola come un nodo in un grafo, GraphSAGE può efficacemente incorporare informazioni dai "vicini", che possono essere altre frasi correlate o parole chiave, per generare una rappresentazione ricca e informativa di ogni nodo.

Inoltre, l'approccio induttivo di GraphSAGE lo rende adattabile ai cambiamenti e alle evoluzioni del contenuto del testo. Poiché non richiede la presenza dell'intero grafo durante la fase di addestramento, è in grado di generare rappresentazioni per nuovi nodi (frasi o parole) che potrebbero apparire in documenti non visti durante la fase di addestramento. Questa caratteristica è particolarmente utile per la Summarization di testi, dove i documenti possono variare notevolmente in termini di contenuto e struttura.

3.4.6 EdgeGCN

Le Graph Convolutional Networks (GCN) hanno dimostrato una notevole capacità nel modellare relazioni tra nodi all'interno di un grafo. Tuttavia, molte implementazioni si concentrano principalmente sui nodi, trascurando in qualche modo le informazioni potenzialmente ricche fornite dagli archi del grafo. EdgeGCN colma questa lacuna sottolineando l'importanza degli archi nella convoluzione del grafo.

Caratteristiche principali di EdgeGCN:

- **Convolution basata sugli archi:** A differenza delle tradizionali GCN che eseguono convoluzioni sui nodi, EdgeGCN introduce una convoluzione basata sugli archi. Ciò significa che le informazioni degli archi vengono utilizzate direttamente per aggiornare i nodi, garantendo che le caratteristiche degli archi siano catturate e utilizzate nel modello.

- **Cattura relazioni complesse:** Gli archi in un grafo rappresentano relazioni tra nodi. Con EdgeGCN, queste relazioni diventano una parte centrale dell'apprendimento, permettendo al modello di catturare relazioni più complesse e sottili tra i nodi.

- **Flessibilità nella definizione degli archi:** Con EdgeGCN, gli archi possono rappresentare qualsiasi tipo di relazione, che si tratti di relazioni spaziali, temporali o semantiche. Questa flessibilità consente a EdgeGCN di essere applicata a una vasta gamma di problemi e dataset.

- **Performance migliorata:** Nei test, EdgeGCN ha dimostrato di fornire prestazioni migliori rispetto alle tradizionali GCN in una serie di compiti, grazie alla sua capacità di utilizzare informazioni dagli archi.

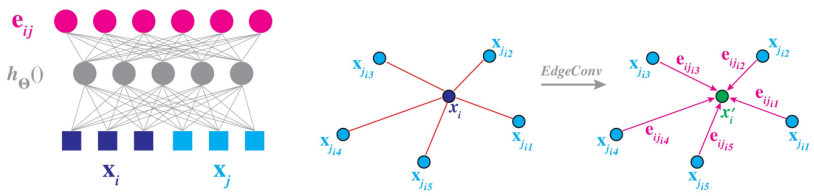

Figura 3.10: Sinistra: Calcolo di una caratteristica dell'arco, e_{ij}, da una coppia di punti, x_i e x_j. In questo esempio, $h_\Theta()$ è realizzato tramite un livello completamente connesso, e i parametri apprendibili sono i pesi associati. Destra: L'operazione EdgeConv. L'output di EdgeConv è calcolato aggregando le caratteristiche degli archi associati a tutti gli archi che partono da ogni vertice connesso. Fonte [27]

Importanza delle EdgeGCN in G-SEEK per la Summarization di Testi

Nel contesto della Summarization di testi, le relazioni tra le diverse parti di un testo sono di fondamentale importanza. La capacità di comprendere e rappresentare queste relazioni può fare la differenza tra una Summarization accurata e una che perde dettagli cruciali o connessioni tra concetti.

EdgeGCN, con il suo focus sull'incorporazione delle informazioni sugli archi, si rivela particolarmente vantaggiosa per G-SEEK nella Summarization dei testi:

- **Cattura Relazioni Semantiche:** Nella Summarization, le relazioni semantiche tra frasi o paragrafi sono cruciali. EdgeGCN, enfatizzando le informazioni sugli archi, può catturare tali relazioni in modo più esplicito, garantendo che le connessioni semantiche tra le parti del testo siano conservate nella Summarization.

- **Rappresentazione Dettagliata:** Mentre le parole all'interno di una frase sono importanti, le relazioni tra le frasi possono spesso fornire un contesto essenziale. EdgeGCN assicura che queste relazioni siano considerate e rappresentate adeguatamente, portando a sommarizzazioni più ricche e dettagliate.

- **Adattabilità:** Dato che EdgeGCN può considerare qualsiasi tipo di relazione rappresentata da un arco, è flessibile nel gestire diverse tipologie di testi con varie strutture e stili. Ciò lo rende adatto a una vasta gamma di compiti di Summarization.

- **Efficienza:** Utilizzando sia informazioni nodali che archi, EdgeGCN può ottenere una rappresentazione più completa del testo in meno passaggi,

potenzialmente accelerando il processo di Summarization e migliorando l'efficienza.

3.5 Analisi degli LLM Utilizzati

Nel corso della ricerca, è stata posta particolare attenzione ai modelli seq2seq, una classe di modelli di trasformazione estremamente potenti nel campo del Natural Language Processing (NLP). I modelli seq2seq sono stati selezionati per la loro capacità di elaborare e generare testo in modo sequenziale, il che li rende ideali per applicazioni come la summarization di testi.

Caratteristiche Principali dei Modelli Seq2Seq Utilizzati:

- **Input IDs (Identificativi di Input):** I modelli seq2seq ricevevano `input_ids`, che rappresentano una sequenza di token convertiti in identificativi numerici. Questi ID vengono utilizzati per mappare il testo originale in una forma che il modello può elaborare.

- **Attention Mask (Maschera di Attenzione):** L'`attention_mask` è utilizzata per indicare al modello quali parti del testo devono essere considerate durante il calcolo dell'attenzione. Questo è particolarmente utile per gestire sequenze di lunghezza variabile e per ignorare i token di padding.

- **Global Attention Mask (Maschera di Attenzione Globale):** Alcuni modelli, in particolare LED (Longformer Encoder-Decoder) e PRIMERA, utilizzano la `global_attention_mask` per enfatizzare determinati token durante il processo di attenzione. La `global_attention_mask` viene impiegata per dare priorità ai token che rappresentano informazioni chiave in un testo lungo, consentendo al modello di concentrarsi su sezioni specifiche del testo mentre mantiene un contesto globale.

- **Labels (Etichette):** Le `labels` venivano usate per fornire al modello il target desiderato durante l'addestramento, come ad esempio il testo sommarizzato corretto. Questo aiuta il modello a imparare a generare il testo desiderato in base all'input fornito.

Risultati e Applicazioni:

I modelli seq2seq si sono dimostrati strumenti versatili e efficaci per affrontare la sfida della summarization automatica. La loro capacità di elaborare

sequenze di testo lunghe e complesse, unita alla possibilità di focalizzare l'attenzione su parti rilevanti del testo, li ha resi particolarmente adatti per analizzare e condensare documenti lunghi in riassunti concisi e informativi.

3.5.1 Pegasus

Il modello PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization) rappresenta una significativa innovazione nel campo della Summarization automatica. Sviluppato da Google Research, questo modello si basa sull'idea di pre-addestrare modelli di Summarization rimuovendo e poi ricreando frasi specifiche dai documenti, invece del tradizionale approccio di mascherare parole o token casuali come in altri modelli di pre-addestramento.

Architettura del Modello PEGASUS

L'architettura di PEGASUS estende il classico modello Transformer, introducendo specifiche ottimizzazioni e modifiche per adattarlo al compito della Summarization abstrattiva. Sebbene mantenga molte delle caratteristiche fondamentali del Transformer, PEGASUS introduce nuovi elementi che lo rendono particolarmente efficace per la Summarization. Ecco una panoramica delle caratteristiche distintive di PEGASUS rispetto al Transformer classico:

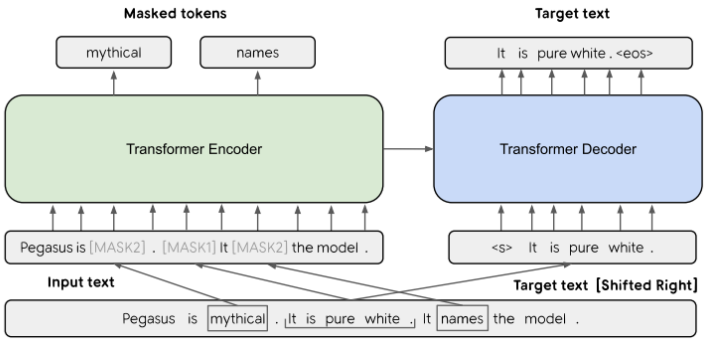

Figura 3.11: L'architettura di base di PEGASUS è un encoder-decoder Transformer standard. Sia GSG che MLM vengono applicati simultaneamente a questo esempio come obiettivi di pre-training. Originariamente ci sono tre frasi. Una frase è mascherata con [MASK1] e usata come testo di generazione target (GSG). Le altre due frasi rimangono nell'input, ma alcuni token vengono mascherati casualmente da [MASK2] (MLM). Fonte [28]

- **Strategia di Pre-addestramento**: La principale distinzione di PEGASUS rispetto ad altri modelli basati su Transformer è la sua strategia di

pre-addestramento. Invece di semplicemente mascherare parole o token casuali, PEGASUS adotta un approccio in cui rimuove intere frasi (gap-sentences) dai documenti e poi addestra il modello a ricreare queste frasi. Questa strategia simula il processo di Summarization, ponendo l'accento sulla generazione di contenuto informativo.

- **Selezionatore di Frasi**: Una componente critica di PEGASUS è il meccanismo che determina quali frasi rimuovere durante la fase di pre-addestramento. Questo selezionatore di frasi è progettato per identificare le frasi più informative, garantendo che il modello si concentri sulle parti del testo che sono più rilevanti per la Summarization.

- **Ottimizzazione per Summarization**: Mentre il Transformer classico è progettato per una varietà di task di NLP, PEGASUS è specificamente ottimizzato per la Summarization. Ciò si riflette in aspetti come la dimensione del modello, il numero di strati, e le specifiche configurazioni di addestramento.

- **Fine-tuning per Task Specifici**: Anche se molti modelli Transformer possono essere adattati per specifici compiti, PEGASUS è particolarmente progettato per il fine-tuning sulla Summarization. Questo consente al modello di ottenere prestazioni di punta su dataset di Summarization specifici.

- **Adattabilità**: A differenza di alcuni modelli Transformer che sono progettati per compiti specifici, PEGASUS è flessibile e può essere adattato per una varietà di compiti di Summarization, sia estrattiva che abstrattiva.

3.5.2 PRIMERA

La Summarization di multi-documenti rappresenta il compito di generare un riassunto partendo da un gruppo di documenti correlati. I metodi all'avanguardia per la Summarization di multi-documenti sono principalmente basati su grafi o sono gerarchici. Benché questi modelli siano efficaci, spesso necessitano di informazioni aggiuntive specifiche del dominio o architetture personalizzate per il dataset, rendendo difficile sfruttare i modelli di linguaggio pre-addestrati. Di conseguenza, modelli pre-addestrati come BERT o GPT potrebbero non essere la scelta ottimale per la Summarization di multi-documenti.

Motivazione e Approccio

PRIMERA è stato proposto come un modello pre-addestrato specificamente progettato per la Summarization di multi-documenti. L'obiettivo principale è

di ridurre la necessità di architetture specifiche per il dataset e grandi quantità di dati etichettati per il fine-tuning. Questo modello è stato concepito per identificare e aggregare informazioni salienti attraverso un "cluster" di documenti correlati durante il pre-addestramento.

L'approccio adottato è l'obiettivo di generazione di frasi gap (GSG), in cui diverse frasi vengono mascherate dal documento di input e poi recuperate in ordine nel decoder. PRIMERA introduce una nuova strategia per la mascheratura delle frasi GSG chiamata "Entity Pyramid", che mira a mascherare le frasi salienti nell'intero cluster e addestrare il modello a generarle.

Architettura del Modello

L'architettura di PRIMERA punta a minimizzare la modellazione specifica del dataset. Per sommare un gruppo di documenti correlati, PRIMERA semplicemente concatena tutti i documenti in una singola sequenza lunga e li elabora attraverso un modello transformer encoder-decoder.

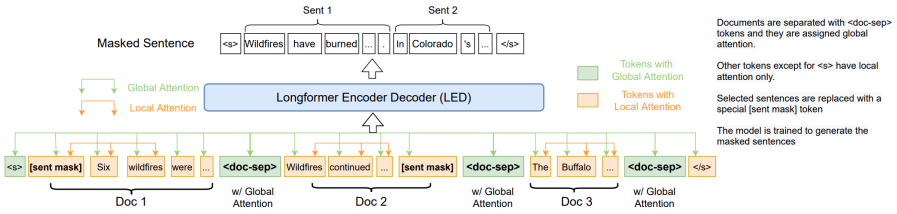

Figura 3.12: Struttura del modello primera. Fonte [29]

Dato l'ampio volume della sequenza concatenata, PRIMERA impiega il modello Longformer-Encoder-Decoder (LED), un transformer efficiente con complessità lineare rispetto alla lunghezza dell'input. LED adotta un meccanismo di attenzione locale+globale sparso nella fase di auto-attenzione dell'encoder, mantenendo però una completa attenzione nel decoder e nell'attenzione incrociata.

Durante la concatenazione, vengono introdotti speciali token separatori di documenti tra i documenti per far sì che il modello riconosca i limiti dei singoli documenti. A questi token viene attribuita un'attenzione globale, permettendo al modello di condividere informazioni attraverso i documenti.

Principali Contributi

1. Introduzione di PRIMERA, il primo modello di generazione pre-addestrato per input multi-documento incentrato sulla Summarization.

2. Proposta di "Entity Pyramid", una nuova strategia di pre-addestramento che guida il modello nella selezione e aggregazione di informazioni rilevanti dai documenti.

3. Ampia valutazione di PRIMERA su 6 dataset di Summarization multi-documento provenienti da 3 diversi domini, dimostrando che supera gli attuali modelli di punta in molte di queste valutazioni.

Entity Pyramid Masking

La tecnica di Pyramid Evaluation si basa sull'intuizione che la rilevanza di una unità informativa può essere determinata dal numero di riassunti di riferimento (ossia standard d'oro) che la includono. Questa unità informativa è denominata Summary Content Unit (SCU) e rappresenta fatti singoli sotto forma di parole o frasi. Gli SCU sono identificati da annotatori umani in ogni riassunto di riferimento e ricevono un punteggio proporzionale al numero di riassunti di riferimento che li contengono. Il Pyramid Score di un riassunto candidato è poi il valore medio normalizzato dei punteggi degli SCU che contiene. Un vantaggio del metodo Pyramid è che valuta direttamente la qualità del contenuto.

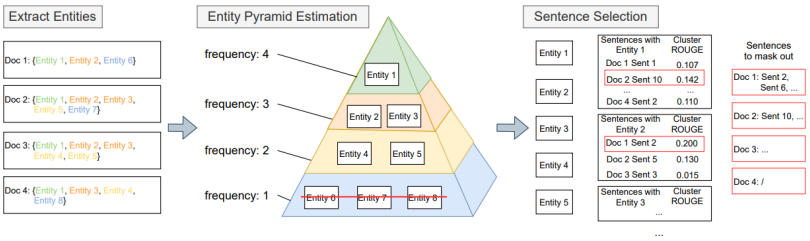

Figura 3.13: La strategia della Piramide delle Entità per selezionare frasi salienti da mascherare. La piramide delle entità si basa sulla frequenza delle entità nei documenti. Le frasi più rappresentative vengono scelte basandosi sul Cluster ROUGE per ciascuna entità con frequenza > 1. Fonte [29]

Inspirandosi a come la rilevanza del contenuto viene misurata nella Pyramid Evaluation, si ipotizza che un'idea simile possa essere applicata nella Summarization di multi-documenti per identificare frasi salienti da mascherare. In particolare, per un cluster con più documenti correlati, più documenti contengono un SCU, più quell'informazione dovrebbe essere rilevante per il cluster. Pertanto, dovrebbe essere considerata per l'inclusione nel pseudo-riassunto nel nostro obiettivo di generazione di frasi mascherate.

Tuttavia, gli SCU nella Pyramid Evaluation originale sono annotati dagli umani, il che non è fattibile per il pre-addestramento su larga scala. Come proxy,

si esplora l'uso delle informazioni espresse come entità nominate, poiché sono elementi chiave nell'estrazione delle informazioni dal testo sugli eventi/oggetti e le relazioni tra i loro partecipanti/parti.

La strategia di mascheratura dell'Entity Pyramid si svolge in tre passaggi:

1. **Estrazione delle entità**: Le entità nominate vengono estratte utilizzando SpaCy.

2. **Stima dell'Entity Pyramid**: Viene costruita una piramide di entità per stimare la rilevanza delle entità in base alla loro frequenza nei documenti, ossia il numero di documenti in cui compare ogni entità.

3. **Selezione delle frasi**: Similmente al framework di valutazione Pyramid, si identificano frasi salienti rispetto al cluster di documenti correlati. Si selezionano le entità che rappresentano meglio l'intero cluster piuttosto che un singolo documento e, successivamente, si selezionano in modo iterativo le entità dalla cima della piramide verso il basso. Infine, all'interno di questo insieme di candidate, si trovano le frasi più rappresentative per il cluster misurando la sovrapposizione del contenuto della frase rispetto ai documenti diversi da quello in cui appare.

La strategia dell'Entity Pyramid favorisce le frasi che sono rappresentative di più documenti nel cluster rispetto alla corrispondenza esatta tra pochi documenti.

3.5.3 BART

BART, acronimo di Bidirectional and Auto-Regressive Transformers, è un modello di linguaggio basato sull'architettura Transformer, sviluppato da Facebook AI. Esso rappresenta un'intrigante coniugazione tra gli approcci auto-regressivi e auto-encoder nel contesto del pre-addestramento di modelli di linguaggio.

- **Trasformazioni Arbitrarie**: Una delle caratteristiche distintive di BART è la sua capacità di gestire trasformazioni arbitrarie del testo. Durante il pre-addestramento, parti del documento vengono corrotte, ad esempio sostituendo span di testo con simboli di maschera. Questa corruzione non necessita di essere allineata con l'output del decoder, permettendo una vasta gamma di trasformazioni rumorose.

- **Encoder Bidirezionale**: Il documento corrotto viene processato da un encoder bidirezionale. Questo encoder è in grado di catturare informazioni sia dal contesto precedente che successivo di ogni parola, fornendo una rappresentazione densa del testo corrotto.

- **Decoder Auto-Regressivo**: Dopo l'encoding, un decoder auto-regressivo calcola la probabilità del documento originale, cercando di ricostruire il testo originale dalla sua versione corrotta. Questa fase di "ricostruzione" è ciò che permette a BART di apprendere trasformazioni complesse sul testo.

- **Fine-tuning**: Per la fase di fine-tuning, un documento non corrotto viene fornito sia all'encoder che al decoder. Le rappresentazioni dallo stato nascosto finale del decoder vengono utilizzate per i compiti downstream. Questo approccio sfrutta la potente capacità di BART di catturare e generare informazioni basate sul contesto.

- **Versatilità e Applicazioni**: Grazie a questa architettura, BART può essere efficacemente adattato a una serie di compiti di elaborazione del linguaggio naturale, dalla Summarization alla risposta alle domande, alla traduzione e oltre.

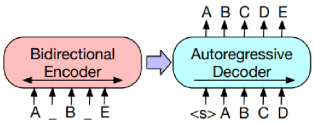

Figura 3.14: Gli input all'encoder non necessitano di essere allineati con gli output del decoder, permettendo trasformazioni arbitrarie di rumore. Qui, un documento è stato corrotto sostituendo porzioni di testo con simboli di maschera. Il documento corrotto (a sinistra) viene codificato con un modello bidirezionale, e poi la probabilità del documento originale (a destra) viene calcolata con un decoder autoregressivo. Per il fine-tuning, un documento non corrotto viene inserito sia nell'encoder che nel decoder, e vengono utilizzate le rappresentazioni dallo stato nascosto finale del decoder. Fonte [30]

3.5.4 LED

Il Longformer Encoder-Decoder (LED) è un modello innovativo specificamente progettato per affrontare la sfida della summarization di documenti lunghi. LED estende l'architettura del Longformer, un modello di trasformazione che introduce un meccanismo di attenzione globale per gestire efficacemente sequenze di lunghezza notevole.

Architettura: La caratteristica distintiva di LED è la sua capacità di elaborare efficacemente testi di lunghezza estesa, superando i limiti tipici dei modelli transformer tradizionali. Questo viene realizzato attraverso l'uso di

un meccanismo di attenzione a finestra scorrevole, che permette al modello di concentrarsi su parti specifiche del testo mantenendo un contesto globale. Inoltre, LED incorpora token di attenzione globale che agiscono come punti di aggregazione per informazioni rilevanti sparse nel testo.

Applicazioni per la summarization: LED si dimostra particolarmente efficace nel contesto della summarization di testi lunghi. La sua architettura consente di elaborare integralmente documenti estesi senza la necessità di troncare il testo. Questo aspetto è fondamentale per garantire che i punti salienti e i dettagli critici del documento siano preservati e considerati nel processo di summarization.

Prestazioni: Come dimostrato nel paper [34], LED mostra prestazioni notevoli nella summarization di documenti estesi, superando modelli precedenti sia in termini di qualità della summarization che di efficienza computazionale. Questo rende LED una scelta ideale per applicazioni in contesti dove la lunghezza dei documenti è un fattore critico, come nella summarization legale, accademica o in report aziendali lunghi.

Capitolo 4

Risultati

4.1 Analisi dei Risultati delle GNN

Come mostrato in Tabella 4.1, sono state sperimentate diverse configurazioni di architettura GNN, tra cui Graph Attention Networks (GAT), Graph Convolutional Networks (GCN), Graph Isomorphism Networks (GIN), DEEPGCN, EDGEGCN, e GRAPHSAGE sul set di validazione di MULTI-LEXSUM (SHORT). Le prestazioni sono state misurate utilizzando tre metriche standard: Precision (P), Recall (R) e Accuracy (A). Ogni architettura ha mostrato un comportamento diverso, suggerendo la necessità di ulteriori indagini per comprendere al meglio il loro potenziale in vari contesti di applicazione.

4.1.1 Metriche di Valutazione

Per valutare le prestazioni delle diverse configurazioni di architettura GNN, sono state impiegate tre metriche standard che forniscono una valutazione complessiva della qualità dei modelli in termini di capacità predittiva e accuratezza:

- **Precision (P)**: Questa metrica calcola la proporzione dei veri positivi rispetto al totale dei positivi predetti dal modello. Una precisione elevata indica una bassa percentuale di falsi positivi.
- **Recall (R)**: Anche conosciuta come sensibilità, la recall misura la proporzione dei veri positivi che sono stati correttamente identificati dal modello sul totale dei veri positivi. Una recall elevata è indicativa di pochi falsi negativi.
- **Accuracy (A)**: L'accuratezza rappresenta la proporzione di previsioni corrette (sia positive che negative) sul totale delle previsioni effettuate. Essa fornisce una misura generale della performance del modello.

L'uso di queste tre metriche insieme fornisce un quadro più completo delle capacità di un modello GNN, consentendo di bilanciare tra la capacità di riconoscere veri positivi (recall) e la capacità di filtrare fuori i falsi positivi (precision), con una valutazione generale dell'efficacia del modello (accuracy). Questo approccio multidimensionale è particolarmente utile quando si confrontano modelli con prestazioni apparentemente simili ma con differenze sostanziali in termini di tipo e frequenza degli errori commessi.

Tabella 4.1: I risultati delle GNN sul set di validazione di MULTI-LEXSUM (SHORT) in diverse configurazioni.

MULTI-LEXSUM(SHORT)																		
GAT			**GCN**			**GIN**			**DEEPGCN**			**EDGEGCN**			**GRAPHSAGE**		
P	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**
38.1	52.5	38.1	35.4	50.3	35.4	32.7	47.7	32.7	39.6	54.7	39.7	39.1	54.2	39.1	37.5	52.5	37.5

La Tabella 4.1 presenta i risultati ottenuti da diverse architetture di Graph Neural Networks (GNN) sul set di validazione MULTI-LEXSUM (SHORT). Ogni colonna rappresenta le metriche di valutazione - Precision (P), Recall (R), e Accuracy (A) - per le differenti architetture GNN testate, che includono Graph Attention Networks (GAT), Graph Convolutional Networks (GCN), Graph Isomorphism Networks (GIN), DEEPGCN, EDGEGCN, e GRAPHSAGE.

Dai risultati si può osservare che DEEPGCN ha ottenuto le prestazioni migliori in termini di Precisione (39.6), Recall (54.7) e Accuracy (39.7). Questo suggerisce che DEEPGCN è più abile nel riconoscere correttamente le parti rilevanti del testo senza includere troppi falsi positivi e nel catturare una maggiore quantità dei veri positivi presenti nei dati.

GAT e EDGEGCN mostrano prestazioni simili e competitive, con GAT che presenta una precisione leggermente inferiore a DEEPGCN, ma una recall comparabile, indicando che è anch'esso capace di identificare molti dei veri positivi.

GCN e GIN hanno mostrato le prestazioni più basse tra le architetture testate, il che potrebbe suggerire che queste architetture potrebbero non essere ottimali per il tipo di dati o il compito specifico considerato nel MULTI-LEXSUM (SHORT). Tuttavia, il loro comportamento merita ulteriori analisi per comprenderne meglio le potenzialità.

GRAPHSAGE si colloca al centro in termini di prestazioni, suggerendo che, mentre non supera DEEPGCN o GAT, rimane un'architettura GNN valida con un buon equilibrio tra precisione e recall.

In generale, queste metriche riflettono l'efficacia con cui ogni modello GNN riesce a elaborare e interpretare le informazioni strutturali e semantiche presenti nei dati. Precisione, recall e accuratezza offrono una visione olistica

delle capacità di ciascun modello, permettendo di valutare come bilanciano il riconoscimento di veri positivi e il filtraggio di falsi positivi, nonché la loro affidabilità generale.

Tuttavia, è importante notare che queste metriche devono essere interpretate con cautela e in relazione alle specifiche del dataset e alle caratteristiche dei diversi modelli GNN. Nei capitoli successivi verranno effettuate ulteriori analisi.

4.2 Analisi dei Risultati

Nella Tabella 4.4 vengono presentati i punteggi ottenuti valutando diverse architetture di modelli su benchmark di dataset per la summarization automatica, denominati Multi-LexSum (Tiny, Short, Long) e GovReport.

4.2.1 Metriche di Valutazione

Nella valutazione delle performance dei modelli di summarizzazione, si sono utilizzate cinque metriche principali per fornire un'analisi completa e multidimensionale delle loro capacità. Ognuna di queste metriche mette in luce aspetti diversi della qualità dei riassunti generati dai modelli, permettendo una valutazione approfondita e sfaccettata delle loro prestazioni. Di seguito, viene fornita una panoramica di ciascuna metrica impiegata.

- **ROUGE-1 (R-1)**: Questa metrica valuta il grado di sovrapposizione tra le parole unigramma nel riassunto generato dal modello e nel riferimento umano. Una punteggio ROUGE-1 elevato indica che il modello è stato in grado di catturare le parole chiave importanti presenti nel testo di riferimento.

- **ROUGE-2 (R-2)**: Simile a ROUGE-1, questa metrica misura la sovrapposizione di bigrammi tra il riassunto generato e il riferimento umano. Un punteggio alto in questa metrica suggerisce che il modello è stato in grado di catturare non solo le parole chiave, ma anche le relazioni di coppia tra le parole, che sono cruciali per il contesto e la coerenza del testo.

- **ROUGE-L (R-L)**: ROUGE-L si concentra sulla lunghezza della massima sotto-sequenza comune tra il riassunto generato e il riferimento umano. È particolarmente utile per valutare la struttura della frase e l'ordine delle parole nel riassunto.

- $\mathcal{R}$: Una metrica composita che combina i punteggi R-1, R-2 e R-L per fornire una valutazione globale delle performance del modello. Questa

metrica fornisce una visione olistica della qualità del riassunto, bilanciando vari aspetti della somiglianza con il riferimento umano.

- **BERTScore (BS)**: Questa metrica utilizza i modelli di linguaggio BERT per valutare la qualità semantica dei riassunti. A differenza delle metriche ROUGE, che si concentrano sulla sovrapposizione lessicale, BERTScore considera il contesto e il significato semantico delle parole, offrendo una valutazione più profonda della coerenza e pertinenza del riassunto rispetto al testo originale.

Tabella 4.2: Il numero di parametri addestrabili dei modelli di linguaggio pre-addestrati (PLM) generativi e la loro dimensione massima di input. Ogni URL inizia con `https://huggingface.co/`. G-SEEK utilizza la lunghezza massima di input del modello downstream ma fornisce frasi salienti anziché troncare quelle eccedenti.

Models	URL	Params	Input
Bart-L	facebook/bart-large	400M	1024
Pegasus-L	google/pegasus-large	568M	1024
Pegasus-XL	google/pegasus-x-large	570M	8192
Led-L	allenai/led-large-16384	459M	8192
Primera-L	allenai/PRIMERA	447M	4096
G-Seek		4M	

I modelli considerati includono variazioni di architetture, con e senza l'uso di una tecnica di enhancement denominata G-Seek e la sua seconda versione, G-Seek-2. I miglioramenti apportati da queste tecniche sono indicati in grassetto e quelli statisticamente significativi, testati con il test t di Student con un valore-p < 0.05, sono segnati con il simbolo †.

Si può notare che l'introduzione di G-Seek e G-Seek-2 ha generalmente migliorato i punteggi su tutte le metriche, segnalando il potenziale di queste tecniche nel migliorare le capacità di summarization dei modelli base.

Questi risultati forniscono una panoramica comprensiva delle prestazioni di modelli avanzati di summarization su datasets di diverse dimensioni, mostrando come l'applicazione di nuove strategie di ottimizzazione possa influenzare significativamente l'efficacia del processo di summarization.

Nella Tabella 4.4, i punteggi di valutazione sono stati ottenuti impiegando diverse combinazioni di architetture di Graph Neural Network (GNN) con vari Language Models (LLM) sui benchmark dei dataset Multi-LexSum (nelle sue versioni Tiny, Short e Long) e GovReport. Questo approccio ha permesso di esplorare l'efficacia di differenti configurazioni GNN nell'ottimizzare le capacità di summarization dei modelli di linguaggio pre-addestrati.

Per ogni Language Model e dataset, abbiamo adottato una specifica architettura GNN che è stata integrata durante il processo di fine-tuning, come segue:

Tabella 4.3: Configurazioni di GNN con Modelli di summarization su diversi Dataset.

	Multi-LexSum (Tiny)	Multi-LexSum (Short)	Multi-LexSum (Long)	GovReport
BART	GraphSAGE	DeepGCN	DeepGCN	GraphSAGE
PEGASUS	GraphSAGE	EdgeGCN	GraphSAGE	GraphSAGE
LED	GraphSAGE	DeepGCN	DeepGCN	DeepGCN
PRIMERA	EdgeGCN	GraphSAGE	DeepGCN	GraphSAGE

Queste combinazioni sono state selezionate sulla base di considerazioni preliminari che suggerivano la loro potenziale complementarità con le specifiche caratteristiche di ciascun modello di linguaggio. L'obiettivo era quello di sfruttare le peculiarità delle diverse topologie di rete GNN per esaltare la qualità della summarization, adattando in modo più efficace i modelli ai contesti dei diversi dataset.

Tabella 4.4: Valutazione dei punteggi sui dataset provati Multi-LexSum (Tiny, Short, Long) e GovReport. L indica "large", rispettivamente. Il miglior punteggio per ciascun modello è evidenziato in grassetto. † indica risultati statisticamente significativi di G-Seek (p-value < 0.05 with student t-test).

	Multi-LexSum (Tiny)					Multi-LexSum (Short)					Multi-LexSum (Long)					GovReport				
Model	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**
Quadratic																				
Bart-l	22.37	7.91	19.74	16.61	76.17	41.45	**18.74**	35.81	31.70	79.89	41.41	16.47	38.98	31.88	79.13	48.46	14.03	44.83	34.94	80.82
w/ G-Seek	24.46	7.70	20.34	17.41	77.07	41.57	16.72	35.78	31.01	79.97	43.92	17.43	41.08	33.67	80.19	51.46	17.12	48.05	37.97	**81.78**
w/ G-Seek-2	**27.72**†	**11.04**†	**22.64**†	**20.49**†	**79.58**†	**42.48**†	15.98	**37.57**†	**31.85**†	**80.77**†	**48.08**†	**22.06**†	**45.57**†	**38.48**†	**82.15**†	**52.3**†	**17.39**†	**48.65**†	**39.45**†	81.77
Pegasus-l	15.09	3.20	12.07	10.09	70.27	38.29	16.31	32.63	28.83	78.58	40.19	16.13	37.82	31.02	78.39	47.12	14.07	44.82	34.55	80.51
w/ G-Seek	19.84	5.13	16.28	13.70	73.12	38.78	16.32	33.26	29.18	79.11	42.38	17.04	39.88	32.68	79.53	50.55	**17.06**	48.01	37.67	**81.30**
w/ G-Seek-2	**23.57**†	**8.85**†	**17.78**†	**16.23**†	**75.23**†	**42.46**†	**17.82**†	**37.77**†	**32.3**†	**80.59**†	**46.79**†	**21.94**†	**43.94**†	**37.56**†	**81.99**†	**50.76**†	16.93	**48.12**†	**38.58**†	81.29
Linear																				
Led-l	22.86	7.98	18.86	16.50	76.20	40.09	17.50	35.15	30.63	79.51	45.26	20.01	42.66	35.52	81.31	53.86	19.53	49.28	39.96	82.65
w/ G-Seek	24.39	7.96	20.55	17.55	77.19	40.95	16.28	35.31	30.51	80.56	45.42	18.87	42.93	35.23	81.03	55.63	21.08	50.67	41.49	82.77
w/ G-Seek-2	**27.74**†	**12.99**†	**22.38**†	**20.94**†	**78.43**†	**46.98**†	**21.08**†	**41.42**†	**36.49**†	**82.7**†	**50.94**†	**25.75**†	**47.58**†	**41.48**†	**83.34**†	**58.29**†	**22.37**†	**54.04**†	**44.87**†	**83.01**†
Primera-l	25.37	8.13	20.84	18.02	76.45	40.20	14.88	34.88	29.63	80.31	45.31	21.06	42.44	35.85	81.34	54.20	19.37	50.20	40.28	79.75
w/ G-Seek	25.76	7.59	21.36	18.13	77.26	**43.99**	18.67	**37.55**	33.02	81.32	45.92	19.61	42.59	35.55	81.36	**57.13**	**21.20**	**53.64**	**42.87**	80.37
w/ G-Seek-2	**27.53**†	**10.92**†	**22.52**†	**20.32**†	**78.01**†	43.65	**20.87**†	30.09	**34.54**†	**82.29**†	**49.51**†	**23.61**†	**46.24**†	**39.63**†	**82.59**†	55.59	20.54	50.84	42.24	**82.54**†

Analisi su Nuovi Domini. La versione avanzata del modello G-SEEK, denominata G-SEEK-2, è stata testata su una varietà di nuovi domini per valutare la sua capacità di generalizzare e la sua robustezza. Nella Tabella 4.5, presentiamo i risultati comparativi ottenuti utilizzando sia i modelli quadratici che lineari su tre dataset distinti: WIKI-CAT-SUM-ANIMAL, MULTI-NEWS e WCEP.

Le prestazioni dei modelli sono state valutate utilizzando le metriche ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L), il punteggio aggregato

ROUGE ($\mathcal{R}$) e il BERTScore (BS), fornendo così una valutazione comprensiva che spazia dalla corrispondenza lessicale alla coerenza del testo generato.

Dai risultati si nota un miglioramento sostanziale nei modelli che integrano G-SEEK-2, evidenziando l'efficacia dell'uso delle GNN nel processo di summarizzazione. In particolare, il modello G-SEEK-2 ha mostrato un incremento nei punteggi ROUGE e BERTScore, suggerendo una maggiore fedeltà al testo di riferimento e una qualità superiore del riassunto prodotto.

Il confronto tra le configurazioni quadratiche e lineari rivela che i modelli lineari potenziati da G-SEEK-2 tendono a fornire prestazioni migliori, come si evince chiaramente nel dataset WCEP, dove G-SEEK-2 raggiunge il punteggio ROUGE aggregato di 39.27 e un BERTScore di 87.28, il più alto tra tutti i modelli testati.

L'adattabilità di G-SEEK-2 a contesti diversi dal suo training originario dimostra la sua capacità di trattare con successo vari tipi di testi, mantenendo un'alta qualità di summarizzazione. Questa versatilità lo rende particolarmente adatto per applicazioni nel mondo reale, dove si richiede la gestione di un'ampia gamma di argomenti e stili di scrittura.

I miglioramenti delle prestazioni, evidenziati dal simbolo † e indicanti risultati statisticamente significativi, sono attribuibili alle novità introdotte in G-SEEK-2.

Tabella 4.5: Valutazione dei punteggi sui dataset provati WIKI-CAT-SUM-ANIMAL, MULTI-NEWS e WCEP. L indica "large", rispettivamente. Il miglior punteggio per ciascun modello è evidenziato in grassetto. † indica risultati statisticamente significativi di G-SEEK (p-value < 0.05 with student t-test).

	WIKI-CAT-SUM-ANIMAL					**MULTI-NEWS**					**WCEP**				
Model	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**	**R-1**	**R-2**	**R-L**	$\mathcal{R}$	**BS**
Quadratic															
BART-L	37.04	**17.76**	**27.24**	**30.1**	78	**43.94**	**13.42**	**20.08**	**32.34**	**79.1**	41.33	**24.89**	**33.72**	**33.87**	83
G-SEEK	45.86	14.49	24.04	27.62	78.29	38.25	10.54	18.83	29.2	78.36	36.51	15.79	26.49	26.93	80.56
G-SEEK-2	**39.64**	14.28	25.38	29.74	**79.20**	40.68	9.97	17.78	28.37	78.21	**41.35**	18.09	29.28	29.75	**84.39**
PEGASUS-X-L	37.45	13.98	23.15	26.45	78.18	33.7	8.42	16.63	23.5	77.58	44.59	25.16	37.04	35.82	82.55
G-SEEK	36.8	13.44	23.33	27.88	78.9	39.63	11.18	17.77	28.13	77.58	46.25	25.17	35.74	35.75	86.17
G-SEEK-2	**37.8**	**14.47**	**24.48**	**28.84**	**79.1**	**44.93**	**13.43**	**19.67**	**32.44**	**79.13**	**48.97**	**27.54**	**41.65**	**39.37**	**85.58**
Linear															
LED-L	39.12	**18.27**	**28.6**	31.57	78.35	44.47	12.54	**19.56**	32.2	**79.3**	51.1	23.9	37.37	37.34	85.01
G-SEEK	40.81	15.72	25.75	**31.6**	78.61	42.92	11.25	18.97	30.14	79.29	45.32	21.83	34.06	33.75	85.85
G-SEEK-2	**40.94**	17.13	26.85	31.2	**79.73**	**44.68**	**13.94**	19.47	**32.35**	78.96	**51.26**	**26.29**	**41.22**	**39.27**	**87.28**
PRIMERA-L	44.18	19.49	28.83	33.16	80	39.9	**9.79**	**18.67**	28.88	78.3	46.24	**26.33**	38.07	**37.15**	84.49
G-SEEK	40.71	16.69	27.58	31.92	79.66	**42.49**	10.34	18	**29.38**	78.41	43.71	25.82	35.33	35.45	84.63
G-SEEK-2	**44.32**	**21.06**	**29.33**	**35.7**	**80.71**	40.2	9.09	17.02	28.18	**78.89**	**48.13**	25.42	**38.11**	36.91	**86.06**

4.3 Impatto Ambientale

Nel contesto attuale, caratterizzato da una crescente sensibilizzazione verso le tematiche ambientali, l'importanza di ridurre i consumi energetici legati alle tecnologie informatiche è sempre più evidente. Il settore del Natural Language Processing (NLP) non fa eccezione, richiedendo l'impiego di considerevoli risorse computazionali che si traducono in significativi consumi energetici e produzione di calore.

La generazione di energia, specialmente da fonti non rinnovabili, è una delle principali cause delle emissioni di gas serra nell'atmosfera. Una riduzione del fabbisogno energetico comporta, di conseguenza, un abbattimento proporzionale delle emissioni inquinanti.

NB: Nel campus universitario di Cesena, si stima la presenza di circa 500 pannelli fotovoltaici, con una potenziale produzione energetica stimata attorno ai 200 kW all'ora. Questa stima, basata su una valutazione approssimativa, sottolinea il potenziale del campus nel contribuire alla riduzione dell'impronta di carbonio attraverso l'utilizzo di energie rinnovabili.

4.3.1 Riduzione dell'inquinamento con il Modello G-SEEK

L'adozione del modello G-SEEK ha portato a una riduzione media del tempo di elaborazione del 20%. Considerando che una scheda grafica come la RTX 3090 ha un consumo energetico di 350 watt e la CPU AMD EPYC 7443 di 200watt, la riduzione del tempo di elaborazione implica una riduzione significativa del consumo energetico. Questa economia di energia non solo si traduce in una minore spesa energetica, ma contribuisce anche a ridurre l'impatto ambientale associato all'utilizzo prolungato di risorse computazionali. Questi dati sottolineano l'importanza di sviluppare soluzioni di machine learning più efficienti dal punto di vista energetico, non solo per ridurre i costi ma anche per mitigare l'impatto ambientale della ricerca e dell'innovazione tecnologica. Analizzando i dati della Tabella 4.5 il modello G-SEEK-2 mostra un miglioramento del BERTScore (BS) rispetto al modello BART-l a parità di tempo di elaborazione e numero di token. In particolare, con 1024 token, il modello G-SEEK-2 raggiunge un BS di 77.87 contro il 76.59 di BART-l, indicando una maggiore qualità del riassunto generato. Anche quando il numero di token è ridotto a 512 o 256, il modello G-SEEK-2 mantiene un vantaggio in termini di BS, suggerendo che il modello è più efficiente nell'identificare e processare le informazioni rilevanti all'interno del testo.

			Multi-LexSum (Tiny)
Model	**token**	**TIME**	**BS**
BART-L	1024	8	76.59
G-SEEK-2	1024	8	77.87
BART-L	512	6	75.33
G-SEEK-2	512	6	77.97
BART-L	256	4	71.48
G-SEEK-2	256	4	75.38

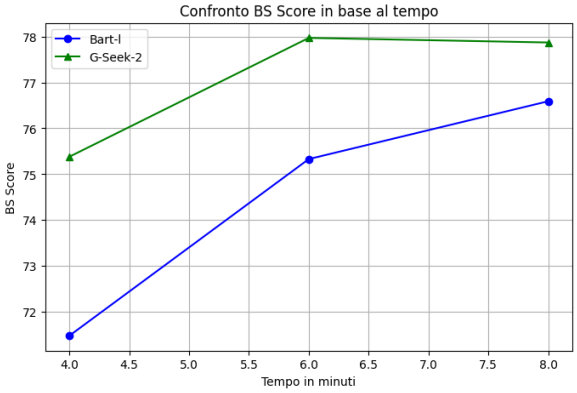

Figura 4.1: Confronto dei punteggi BERTScore (BS) per i modelli BART-L e G-SEEK-2 su diverse lunghezze di token nel dataset MULTI-LEXSUM (TINY). Le prestazioni sono misurate in termini di tempo di elaborazione (TIME) e qualità della summarization (BS). A sinistra è presentata la tabella con i risultati numerici e a destra la rappresentazione grafica dei dati.

Questa efficienza si riflette direttamente in termini di consumo energetico. Considerando un'operatività costante e un numero elevato di elaborazioni, l'impiego di G-SEEK-2 potrebbe tradursi in un risparmio energetico significativo, con impatti positivi sull'impronta di carbonio complessiva del processo di summarization. In un contesto di crescente attenzione verso la sostenibilità e l'efficienza energetica, l'impiego di G-SEEK-2 rappresenta un passo importante verso un'informatica più verde. Considerando un consumo totale di 550 watt e un fattore di emissione di carbonio di 0,432 kg/kWh [35] per un training di 200 ore, si stima la seguente riduzione dell'impronta di carbonio:

- Emissioni totali di CO_2 senza G-SEEK:
  $200\,\text{ore} \times 550\,\text{watt} \times 0,432\,\text{kg/kWh} = 47,52\,\text{kg di } CO_2$
- Emissioni totali di CO_2 con G-SEEK (20% di tempo in meno):
  $0,8 \times 47,52\,\text{kg di } CO_2 = 38,016\,\text{kg di } CO_2$
- Risparmio totale in CO_2:
  $47,52\,\text{kg di } CO_2 - 38,016\,\text{kg di } CO_2 = 9,504\,\text{kg di } CO_2$

4.3.2 Impatto Ambientale dei componenti dei sistemi HPC

Si è esaminato l'impatto relativo in termini di carbonio incorporato di vari componenti dei sistemi HPC. I componenti analizzati sono dettagliati nella

figura 4.2 e compaiono frequentemente nella lista dei top 500 supercomputer. Rappresentano una vasta gamma in termini di fornitori e periodi e possiedono specifiche relative al carbonio accessibili o derivabili.

Nella figura 4.2, si confronta l'impatto carbonico delle componenti GPU e CPU. La figura 4.2 indica che i dispositivi GPU presentano un impatto carbonico fino a 3.4 volte superiore rispetto ai dispositivi CPU. Tuttavia, quando si normalizza l'impatto carbonico rispetto alle prestazioni dell'operazione FP64, la tendenza si inverte. Infatti, ogni dispositivo CPU ha un impatto carbonico per FLOPS superiore rispetto a qualsiasi dispositivo GPU. Questo è dovuto al fatto che, nonostante un impatto carbonico minore, le CPU offrono prestazioni significativamente inferiori.

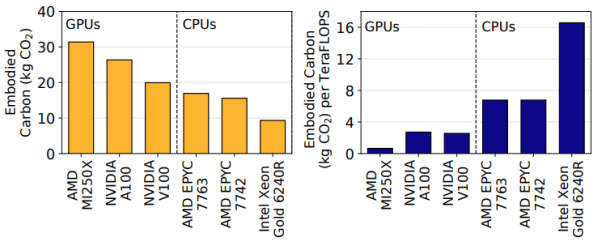

Figura 4.2: Impronta di carbonio incorporata dei dispositivi GPU/CPU e l'impronta normalizzata in base alla prestazione teorica in virgola mobile a doppia precisione (TeraFLOPS). Fonte [31]

L'analisi ha rivelato che le GPU tendono ad avere un impatto carbonico nettamente superiore rispetto alle CPU. Nonostante un impatto carbonico complessivo maggiore, quando si considera l'impatto carbonico normalizzato alle prestazioni grezze (gCO2/FLOPS), le GPU risultano più efficienti rispetto alle CPU.

4.3.3 Analisi del Carbonio in Relazione al Numero di GPU

Nella 4.3, si osserva come l'impatto carbonico e le prestazioni del sistema cambiano al variare del numero di GPU in un nodo. Le prestazioni del sistema con 1-GPU, 2-GPU e 4-GPU sono state caratterizzate utilizzando tre set di benchmark presenti nella figura 4.2. Le prestazioni sono state confrontate con l'impatto carbonico del nodo. Come atteso, l'aumento dell'impatto carbonico è proporzionale al numero di GPU aggiunte. Quando il numero di GPU viene incrementato a 2, sia l'impatto carbonico che le prestazioni del nodo crescono di circa il 30% al 40% rispetto all'impatto carbonico normalizzato e

alle prestazioni corrispondenti. Tuttavia, aumentando ulteriormente il numero di GPU a 4, l'incremento delle prestazioni non riesce a tenere il passo con l'aumento dell'impatto carbonico a causa dell'overhead di comunicazione tra le GPU. Di conseguenza, il rapporto prestazioni-impatto carbonico diminuisce, attestandosi a circa 0,88 per i benchmark NLP e CANDLE e a 0,79 per i benchmark Vision.

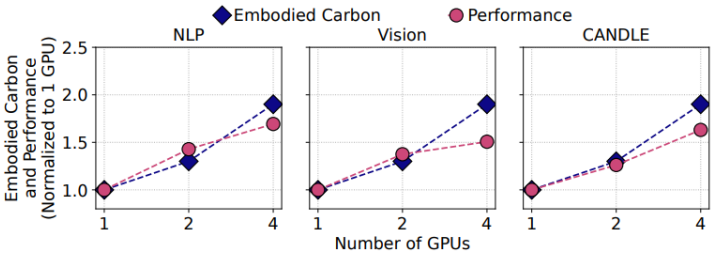

Figura 4.3: L'aumento del numero di GPU in un nodo può migliorare le prestazioni del nodo, ma porta anche a un'impronta di carbonio incorporata più elevata. Tuttavia, come previsto, i guadagni di prestazione tendono a stabilizzarsi, ma l'impronta di carbonio continua ad aumentare. Fonte [31]

Gli esperimenti dimostrano che l'impatto carbonico per unità di prestazioni di un carico di lavoro può peggiorare con l'aumento del numero di GPU. Sebbene le prestazioni potrebbero non crescere linearmente con l'incremento delle GPU, l'impatto carbonico lo fa, portando a un aumento dell'impatto carbonico per unità di prestazione.

4.3.4 Impatto Ambientale dei Large Language Models

Negli ultimi anni, l'evoluzione dei Large Language Models (LLM) ha portato a una crescente necessità di risorse computazionali, con conseguenti implicazioni ambientali. Questi modelli, pur offrendo risultati eccezionali in termini di prestazioni, hanno un costo significativo in termini di consumo energetico e produzione di emissioni di carbonio.

Carburacy [32] ideato dai ricercatori Gianluca Moro, Luca Ragazzi e Lorenzo Valgimigli, emerge come una soluzione innovativa per affrontare questa sfida. È la prima metrica che tiene conto sia delle prestazioni del modello (efficacia) che del suo impatto ambientale (costo in termini di emissioni di carbonio). L'obiettivo di Carburacy è fornire una valutazione equilibrata che consideri sia la qualità del modello sia l'ecosostenibilità.

La formula di Carburacy, presentata come:

$$\Upsilon = e^{\frac{\log_{\alpha} R}{1+C\cdot\beta}}$$

dove α e β sono iperparametri, rappresenta un equilibrio tra l'efficacia R e il costo C del modello. In particolare, α regola la curvatura della funzione di efficacia, premiando i guadagni di efficacia che sono matematicamente più significativi. Mentre β pondera il costo del modello, assegnando più o meno importanza all'impronta di carbonio rispetto alle prestazioni.

Vantaggi di Carburacy:

- **Promuove l'Ecosostenibilità**: Carburacy incoraggia lo sviluppo e l'adozione di modelli che non solo siano efficaci, ma anche ecologicamente sostenibili.
- **Equilibrio tra Prestazioni e Costo Ambientale**: Offre una metrica comprensiva che tiene conto sia delle prestazioni del modello sia del suo impatto ambientale, permettendo una valutazione più olistica.

4.3.5 Analisi sul Numero di Campioni di Addestramento

La Figura 4.4 illustra i punteggi di Carburacy su GOVREPORT alimentando i modelli con diverse dimensioni di input e istanze di addestramento.

Il trend di Carburacy non è lineare, ma segue una curva con un picco intorno ai 100 documenti di addestramento. Un comportamento simile può essere osservato da quasi tutti i modelli testati su diverse configurazioni e set di dati (questi risultati sono mostrati nell'Appendice per motivi di spazio).

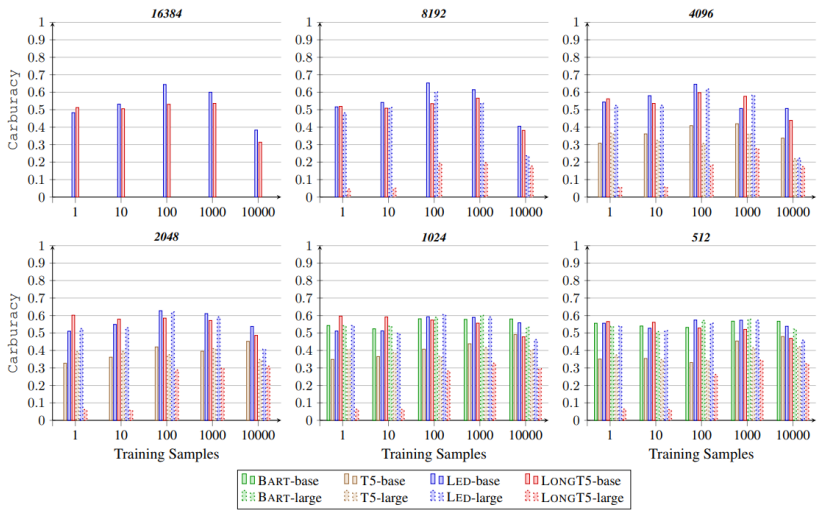

Figura 4.4: I punteggi di Carburacy su GOVREPORT variando la dimensione dell'input e i campioni di addestramento. Fonte [32]

Questa tendenza è attribuibile alle capacità di apprendimento few-shot dei modelli linguistici, che raggiungono buoni risultati con costi molto bassi. Infatti, migliori sono le capacità di apprendimento few-shot, più ecosostenibili sono i modelli, in quanto possono raggiungere un'efficacia notevole con emissioni minime di CO_2. Con un numero maggiore di campioni di addestramento, il costo aggiuntivo non giustifica il leggero miglioramento dell'efficacia, portando quindi a una diminuzione di Carburacy. I modelli di grandi dimensioni, che contengono più parametri, tendono ad ottenere un punteggio di Carburacy simile ai modelli base, anche se producono significativamente più CO_2. Tuttavia, si rivelano inefficienti per sessioni di addestramento prolungate perché emettono più CO_2 senza migliorare significativamente l'efficacia. Infatti, con 10.000 campioni, i modelli base raggiungono un punteggio di Carburacy migliore rispetto alle loro versioni più grandi.

4.3.6 Analisi sulla Dimensione dell'Input

La Figura 4.5 mostra che l'aumento dei token di input per l'addestramento porta a miglioramenti dell'efficacia molto più significativi rispetto alle emissioni di carbonio. A causa della natura del compito, che è la Summarization di documenti lunghi, estendere i token di input significa fornire al modello più informazioni. Pertanto, lo stesso modello nelle stesse impostazioni ma con più token di input genera un riassunto di qualità superiore. Al contrario, un elevato numero di token di input fa sì che il modello consumi più energia e memoria GPU; tuttavia, l'efficacia aumenta notevolmente, incrementando il punteggio complessivo di Carburacy. Il punteggio di Carburacy per lunghezze di input estese è, in media, migliore rispetto a lunghezze più corte. Questo compromesso rende i trasformatori lineari più ecosostenibili nella Summarization di documenti lunghi rispetto a modelli limitati all'elaborazione di pochi token, come BART.

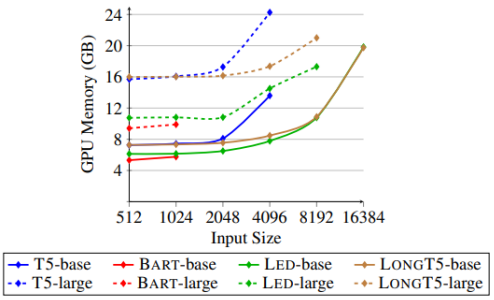

Figura 4.5: L'occupazione della memoria GPU su GOVREPORT variando la dimensione dell'input durante il tempo di addestramento. Fonte [32]

4.3.7 Verso la Carbon Neutrality

Nell'era attuale dell'industrializzazione e della digitalizzazione, l'importanza di un approccio sostenibile e rispettoso dell'ambiente è diventata centrale per molte aziende. Una delle mete più ambite è raggiungere la "carbon neutrality", ovvero un bilancio in cui le emissioni di carbonio prodotte sono bilanciate o azzerate attraverso varie strategie di compensazione.

Le aziende con data center sono tra le principali consumatrici di energia elettrica nel mondo. Questi centri elaborano enormi quantità di dati e richiedono un funzionamento ininterrotto, il che si traduce in un elevato consumo di energia. Per ridurre il proprio impatto ambientale, molte di queste aziende stanno ora investendo in fonti di energia rinnovabile.

Sul fronte dell'hardware, la sostenibilità è diventata un aspetto fondamentale nella produzione di dispositivi e componenti. Molte aziende stanno adottando pratiche ecocompatibili nella produzione, come l'uso di materiali riciclati o facilmente riciclabili, riducendo l'uso di sostanze tossiche e implementando processi di produzione ad alta efficienza energetica. L'hardware prodotto in questo modo non solo ha un impatto ambientale ridotto durante la sua produzione, ma è anche progettato per avere un ciclo di vita più lungo e per essere smaltito in modo responsabile alla fine del suo utilizzo. Questi sforzi rappresentano un passo significativo verso una produzione di hardware più sostenibile e rispettosa dell'ambiente, riducendo ulteriormente l'impronta di carbonio dell'industria tecnologica.

Tuttavia, l'adozione di energie rinnovabili o il riciclaggio al 100% non è sempre possibile ovunque, a causa di limitazioni geografiche, climatiche o infrastrutturali. In questi casi, le aziende adottano altre strategie per compensare le loro emissioni. Una delle pratiche più comuni è la riforestazione o la piantagione di alberi. Gli alberi assorbono il carbonio atmosferico durante la fotosintesi, funzionando come "serbatoi" naturali di carbonio. Investendo nella piantagione di alberi, le aziende non solo compensano le loro emissioni ma contribuiscono anche a rafforzare la biodiversità, prevenire l'erosione del suolo e sostenere le comunità locali attraverso progetti di riforestazione.

4.3.8 Un passo verso la Carbon Neutrality

Nel panorama degli elettronici di consumo, Apple si è distinta per la sua crescente attenzione alle questioni ambientali. Uno degli esempi più emblematici

di questo impegno è l'introduzione dell'Apple Watch Series 9, dichiarato come il primo dispositivo dell'azienda ad essere completamente carbon neutral.

Produzione Sostenibile

La produzione dell'Apple Watch Series 9 tiene conto dell'intero ciclo di vita del prodotto. Apple ha optato per l'utilizzo di materiali riciclati e rinnovabili, riducendo significativamente le emissioni di carbonio associate all'estrazione e alla produzione. Ad esempio, l'alluminio utilizzato per la cassa dell'orologio proviene da fonti riciclate, riducendo così la necessità di nuova estrazione.

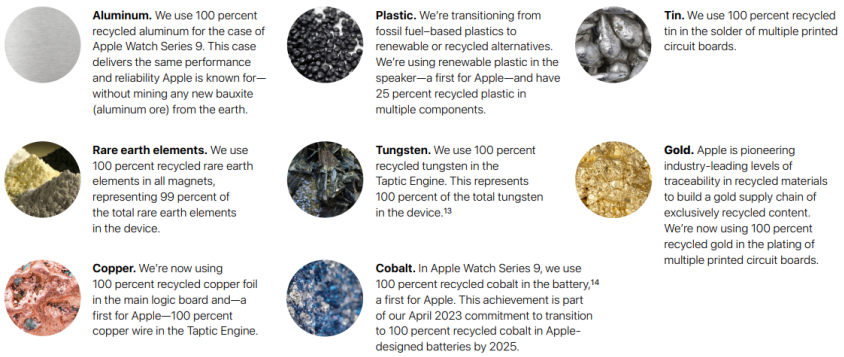

Figura 4.6: I materiali sorgente dell'Apple Watch Series 9 contengono il 30 percento o più di contenuto riciclato o rinnovabile. Fonte [33]

Energia Rinnovabile

Oltre agli sforzi nella fase di produzione, Apple ha assicurato che tutti i suoi stabilimenti di produzione funzionino esclusivamente con energia rinnovabile. Ciò significa che l'energia utilizzata per assemblare l'Apple Watch Series 9 proviene da fonti come il solare, l'eolico e l'idroelettrico, minimizzando ulteriormente le emissioni di carbonio.

Compensazione delle Emissioni

Nonostante gli sforzi per ridurre le emissioni alla fonte, alcune emissioni sono inevitabili. Per compensare ciò, Apple investe in progetti di riforestazione e in altre iniziative volte a rimuovere il carbonio dall'atmosfera. Questi progetti non solo compensano le emissioni di carbonio, ma offrono anche benefici aggiuntivi, come la conservazione della biodiversità e il sostegno alle comunità locali.

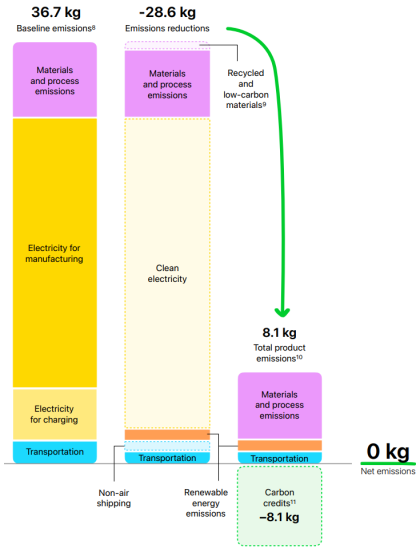

Figura 4.7: Composizione delle Emissioni del apple watch. Fonte [33]

Impegno a Lungo Termine

La decisione di rendere l'Apple Watch Series 9 carbon neutral non è un'azione isolata. Fa parte di un impegno più ampio dell'azienda per garantire che tutti i suoi prodotti e la sua intera catena di approvvigionamento siano carbon neutral entro il 2030. Ciò rappresenta un segnale forte nel settore elettronico di consumo e pone Apple come leader nell'adozione di pratiche sostenibili.

Conclusioni e sviluppi futuri

Questa tesi ha intrapreso un viaggio esplorativo attraverso le frontiere del Natural Language Processing (NLP), partendo dalla solida base fornita dal modello G-SEEK (che è progettato per migliorare le prestazioni nella summarization di documenti basandosi sul riconoscimento delle parti salienti di un testo per poter generare riassunti che catturino i punti essenziali del contenuto originale utilizzando Graph Neural Networks per analizzare e interpretare la struttura e le relazioni semantiche presenti nel testo) per sviluppare la sua seconda iterazione, G-SEEK-2. Sfruttando le avanzate capacità di modelli seq2seq, la ricerca ha approfondito l'uso delle Graph Neural Networks (GNN) e dei Large Language Models (LLM), due componenti fondamentali che hanno permesso di ampliare notevolmente la precisione nella summarization di documenti.

G-seek-2 ha mostrato miglioramenti tangibili nelle metriche di valutazione standard del settore, con un incremento significativo del 18% nel punteggio ROUGE e un avanzamento del 3% nel BERT Score. Questi risultati sottolineano il successo dell'approccio adottato e aprono nuove prospettive per future ricerche e applicazioni pratiche. Parallelamente ai vantaggi in termini di precisione e coerenza dei riassunti generati, G-SEEK-2 ha contribuito a un impatto ambientale più sostenibile. Grazie alla riduzione del 20% nei tempi di training, si è verificata una corrispondente diminuzione del consumo energetico e, di conseguenza, delle emissioni di carbonio associate.

Il nucleo di questa evoluzione è stato l'integrazione di domini nuovi, che ha permesso di valutare e dimostrare la flessibilità e l'adattabilità del modello G-SEEK-2.

Nel contesto ambientale attuale, è anche fondamentale considerare l'impatto energetico e sostenibile di queste tecnologie. L'efficienza energetica e la riduzione dell'impronta di carbonio sono diventate priorità imprescindibili, e i futuri sviluppi nel campo dell'NLP dovranno tener conto di questi aspetti, orientandosi verso soluzioni sempre più verdi.

Questa tesi ha posto una particolare enfasi sull'efficienza energetica e sulla sostenibilità delle soluzioni di NLP. Riconoscendo l'urgenza di operare in maniera eco-responsabile, l'innovativo modello G-SEEK-2 è stato sviluppato

non solo con l'intento di migliorare le prestazioni in termini di precisione e coerenza del testo, ma anche di ridurre l'impronta di carbonio associata all'addestramento e all'utilizzo dei Large Language Models.

Attraverso G-seek-2, si è riusciti a tagliare i consumi energetici di circa il 20% rispetto ai tradizionali approcci NLP, senza compromettere l'efficacia del modello. Questo passo avanti non solo conferma la fattibilità di un NLP più verde, ma stabilisce un nuovo standard per la ricerca futura, che dovrà sempre più spesso bilanciare l'innovazione tecnologica con l'etica ambientale.

In vista degli sviluppi futuri, questo principio di sostenibilità sarà un criterio guida per l'esplorazione di nuove architetture di modelli e algoritmi. L'obiettivo sarà di mantenere l'equilibrio tra un avanzamento tecnologico rapido e la necessità di operare in modo responsabile verso il nostro pianeta, perseguitando strategie che favoriscano l'uso efficiente delle risorse e minimizzino l'impatto ecologico.

In conclusione, il lavoro svolto rappresenta un passo importante verso la comprensione e l'applicazione pratica dei modelli di NLP nel mondo reale. I sviluppi futuri prevedono un ulteriore approfondimento delle tecniche di GNN e LLM, con una particolare attenzione rivolta all'esplorazione di nuovi modelli transformer come LLaMA e Mistral, e al loro potenziale impatto in vari ambiti applicativi. L'obiettivo sarà sempre quello di spingere i confini della tecnologia per ottenere sistemi di NLP ancora più performanti, efficienti e versatili.

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine al mio relatore Prof. Gianluca Moro, la cui guida esperta e il supporto costante hanno reso possibile questo lavoro. La sua passione e la sua conoscenza hanno notevolmente influenzato il mio percorso accademico e hanno alimentato il mio crescente interesse per il campo del Natural Language Processing.

Un ringraziamento va inoltre ai miei correlatori, Lorenzo Valgimigli e Luca Ragazzi, per i loro preziosi consigli e il loro sostegno.

Un grazie va anche ai miei compagni di università, con i quali ho condiviso molte sfide e successi. Le discussioni, i momenti di studio e il supporto reciproco sono stati fondamentali per la mia crescita personale e accademica.

Desidero inoltre ringraziare i miei genitori, per il loro sostegno e la fiducia che hanno sempre avuto in me.

Infine, ringrazio tutti coloro che, anche se non menzionati esplicitamente, hanno contribuito in qualche modo al mio percorso di studi e alla realizzazione di questa tesi.

Bibliografia

[1] Gianluca Moro, Luca Ragazzi, and Lorenzo Valgimigli. Graph-based abstractive summarization of extracted essential knowledge for low-resource scenarios. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 1747–1754. IOS Press, 2023.

[2] SemiEngineering. 3d transistors, 2023. `https://semiengineering.com/knowledge_centers/integrated-circuit/transistors/3d/`.

[3] Kazi Asifuzzaman, Mohamed Abuelala, Mohamed Hassan, and Francisco J. Cazorla. Demystifying the characteristics of high bandwidth memory for real-time systems. In *ICCAD*, pages 1–9. IEEE, 2021.

[4] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. Pioneering chiplet technology and design for the AMD epyc™ and ryzen™ processor families : Industrial product. In *ISCA*, pages 57–70. IEEE, 2021.

[5] Wikipedia. Amdahl's law — Wikipedia, the free encyclopedia, 2023. `https://en.wikipedia.org/wiki/Amdahl%27s_law`.

[6] Abd Al-Imran. Intuition of gradient descent for machine learning, 2023. `https://abdalimran.medium.com/intuition-of-gradient-descent-for-machine-learning`.

[7] Ashwath Salimath. How to use the learning rate finder in tensorflow, 2019. `https://medium.com/octavian-ai/how-to-use-the-learning-rate-finder-in-tensorflow`.

[8] MathWorks. Overfitting, 2023. `https://it.mathworks.com/discovery/overfitting.html`.

[9] Suvodeep Sinha. all about convolutions, 2020. `https://medium.com/nerd-for-tech/all-about-convolutions`.

[10] Applied Intelligence. Dropout, 2023. https://www.researchgate.net/figure/Schematic-layout-of-Dropout33_fig5_371907912.

[11] MathWorks. Convalida incrociata (cross validation), 2023. https://it.mathworks.com/discovery/cross-validation.html.

[12] Luke Shannon-Hill. What is a neural network anyway?, 2018. https://medium.com/@lukeshannonhill/what-is-a-neural-network-anyway.

[13] NVIDIA. What is transfer learning?, 2 2019. https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/.

[14] IBM. Recurrent neural networks, 2023. https://www.ibm.com/it-it/topics/recurrent-neural-networks.

[15] Nvidia. What are graph neural networks?, Oct 2022. https://blogs.nvidia.com/blog/2022/10/24/what-are-graph-neural-networks/.

[16] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. In *EMNLP/IJCNLP (1)*, pages 3442–3448. Association for Computational Linguistics, 2019.

[17] Bishal Bose. Nlp text encoding: Word2vec, 2023. https://medium.com/analytics-vidhya/nlp-text-encoding-word2vec.

[18] Jay Alammar. The illustrated word2vec, 2023. http://jalammar.github.io/illustrated-word2vec/.

[19] Jay Alammar. Visualizing neural machine translation mechanics of seq2seq models with attention, 2018. https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq/models-with-attention/.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[21] Jay Alammar. The illustrated transformer, 2018. https://jalammar.github.io/illustrated-transformer/.

[22] Automatic mixed precision, 2023. https://developer.nvidia.com/automatic-mixed-precision.

[23] Nico Gasparini. Introduzione a docker, 2019. https://nicogaspa.medium.com/introduzione-a-docker-d61f2b46d84c.

[24] Ashki. Cluster analysis and its significance to big data, 2023. https://ashki23.github.io/cluster.html.

[25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR (Poster)*. OpenReview.net, 2018.

[26] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

[27] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[28] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR, 2020.

[29] Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. PRIMERA: pyramid-based masked sentence pre-training for multi-document summarization. In *ACL (1)*, pages 5245–5263. Association for Computational Linguistics, 2022.

[30] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880. Association for Computational Linguistics, 2020.

[31] Baolin Li, Rohan Basu Roy, Daniel Wang, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Toward sustainable HPC: carbon footprint estimation and environmental implications of HPC systems. In *SC*, pages 19:1–19:15. ACM, 2023.

[32] Gianluca Moro, Luca Ragazzi, and Lorenzo Valgimigli. Carburacy: Summarization models tuning and comparison in eco-sustainable regimes with a novel carbon-aware accuracy. In *AAAI*, pages 14417–14425. AAAI Press, 2023.

[33] Apple Inc. Carbon neutral apple watch series 9 product environmental report, Sep 2023. https://www.apple.com/environment/pdf/products/watch/Carbon_Neutral_Apple_Watch_Series_9_PER_Sept2023.pdf.

[34] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

[35] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.