
aracne

Classificazione Decimale Dewey:

006.37 (23.) INTELLIGENZA ARTIFICIALE. VISIONE DA PARTE DELL'ELABORATORE

GERARDO IOVANE

LARGE LANGUAGE MODELS WITH FRAMEWORKS AND PROMPT ENGINEERING IN ARTIFICIAL INTELLIGENCE

aracne

ISBN
979-12-218-2252-6

PRIMA EDIZIONE
ROMA 15 OTTOBRE 2025

Indice

Parte I: Large Language Models 11
1. Introduzione 11
1.1 Evoluzione dei modelli linguistici 13
1.2 Dalla statistica al deep learning 15
1.3 Il Transformer e il paper 'Attention is All You Need' 18
1.4 Perché i modelli sono 'large' e come scalano 20
1.5 Impatto degli LLM su ricerca e sviluppo 21
2. Fondamenti matematici e computazionali 25
2.1 Self-attention: formulazione e calcolo 25
2.2 Positional encoding e gestione del contesto 27
2.3 Matrici di pesi e complessità computazionale 29
2.4 Scaling laws e limiti teorici 30
2.5 Trade-off tra parametri, memoria e velocità 32
3. Training, fine-tuning e RLHF 37
3.1 Dataset e data curation per LLM 37
3.2 Tokenizzazione e embedding 40
3.3 Pre-training vs fine-tuning completo 43
3.4 Tecniche LoRA e QLoRA 45
3.5 Reinforcement Learning from Human Feedback (RLHF) 47
4. Framework per sviluppatori 51
4.1 Hugging Face Transformers in dettaglio 51
4.2 Acceleratori e ottimizzazioni (DeepSpeed, FlashAttention, vLLM) 53
4.3 Pipeline MLOps (MLflow, Weights & Biases) 56
4.4 Librerie per RAG e agenti autonomi 58
5. Deployment e ottimizzazione 63
5.1 Self-hosted vs API cloud 63
5.2 Quantizzazione e modelli leggeri (8-bit, 4-bit) 65
5.3 Containerizzazione con Docker e Kubernetes 67

5.4 Infrastruttura hardware: GPU, TPU ed edge computing 69
5.5 Riduzione dei costi e inferenza efficiente 71
6. Benchmark e comparazione multiparametrica 75
6.1 Metriche: accuratezza, velocità, costi 75
6.2 Benchmark open (AlpacaEval, HELM, MT-Bench) 77
6.3 Radar chart e istogrammi di confronto tra modelli 79
6.4 Analisi costi/benefici 82
6.5 Linee guida per la scelta del modello 85
7. Applicazioni avanzate per sviluppatori 89
7.1 Agent framework (LangChain, LlamaIndex) 89
7.2 RAG con database vettoriali (Pinecone, Weaviate, ChromaDB) 91
7.3 Automazione di workflow complessi e integrazione API 94
7.4 Costruzione di applicazioni LLM-driven 97
7.5 Casi d'uso reali e best practice 99
8. Etica, sicurezza e regolamentazione 103
8.1 Bias, allucinazioni e mitigazioni 103
8.2 Sicurezza in API e modelli self-hosted 105
8.3 Attacchi avversari e protezione modelli 108
8.4 Licenze, AI Act e compliance 112
8.5 Impatti sociali e futuri regolamenti 114
9. Il futuro degli LLM per sviluppatori 119
9.1 Modelli multimodali, specialistici e ibridi 119
9.2 Auto-LLM e modelli evolutivi 121
9.3 Integrazione con AGI e sistemi cognitivi 124
9.4 Tendenze hardware e software emergenti 125
9.5 Roadmap per sviluppatori e scienziati 128
10. Aspetti accessori 131
10.1 Glossario matematico e tecnico 131
10.2 Codice base per ogni LLM presentato 134
10.3 Checklist per deployment e ottimizzazione 136
10.4 Tabelle comparative di versioni, parametri e costi 138
11. Conclusione Parte I 143
Bibliografia Parte I 147
Parte II: Frameworks per LLM 159
12. Introduzione ai Frameworks per l'AI 159

13. GPT (OpenAI) 161
13.1 Architettura GPT-3.5, GPT-4 e GPT-4o 161
13.2 API e modelli function-calling 164
13.3 Strumenti per agenti (Assistants API, plugin, tool calling) 167
13.4 Codice pratico per automazione e integrazione 171
13.5 Best practice per sviluppatori avanzati 174
14. Claude (Anthropic) 179
14.1 Filosofia e architettura Claude 3 Opus/Sonnet 179
14.2 API e differenze rispetto a GPT 181
14.3 Controllo sicurezza e allineamento AI 183
14.4 Codice pratico e use case avanzati 185
14.5 Limiti e prospettive future 187
15. LLaMA (Meta) 191
15.1 LLaMA 2 e LLaMA 3: modelli open source 191
15.2 Fine-tuning personalizzato con Hugging Face 193
15.3 Deployment locale e RAG con LLaMA 196
15.4 Codice e ottimizzazione per hardware consumer 198
15.5 Benchmark rispetto a GPT e Claude 201
16. Falcon (TII) 203
16.1 Modelli Falcon 40B e 180B 203
16.2 Ottimizzazioni per inferenza rapida 205
16.3 Deployment self-hosted 208
16.4 Codice pratico e casi d'uso enterprise 210
16.5 Vantaggi e svantaggi 213
17. Mistral & Mixtral (Mistral AI) 217
17.1 Mistral 7B e Mixtral MoE (Mixture of Experts) 217
17.2 Come funziona MoE per ridurre costi e aumentare velocità 219
17.3 Deployment locale e su cloud 221
17.4 Codice e pipeline pratiche 224
17.5 Confronto con altri modelli leggeri 227
18. Gemini (Google DeepMind) 231
18.1 Gemini 1.5 Pro e Ultra: multimodalità avanzata 231
18.2 API, funzioni e limiti attuali 233
18.3 Codice di integrazione multimodale 235
18.4 Casi d'uso enterprise 237
18.5 Differenze rispetto a GPT e Claude 239

19. Command R+ (Cohere) 243
19.1 Modelli ottimizzati per RAG e knowledge retrieval 243
19.2 API specializzate per document intelligence 245
19.3 Codice pratico per knowledge worker 247
19.4 Integrazione con database vettoriali 249
19.5 Punti di forza e criticità 252
20. Kimo K2 255
20.1 Architettura e innovazioni di Kimo K2 255
20.2 Strumenti e librerie per sviluppatori 257
20.3 Codice pratico e deployment 260
20.4 Confronto prestazioni vs GPT-4o e Claude 261
20.5 Potenzialità future 264
21. MAS (Multi-Agent Systems) con Kimo K2 269
21.1 Architettura MAS aziendale e interazione con LLM 269
21.2 Integrazione di Kimo K2 nel MAS 270
21.3 Caso d'uso: ottimizzazione dei processi aziendali 272
21.4 Codice pratico e interfaccia Kimo K2-MAS 274
21.5 Best practice e limiti dell'approccio LLM-MAS 278
22. Conclusione Parte II 283
Bibliografia Parte II 287
Parte III: Prompt Engineering in Emerging Artificial Intelligence 299
23. Introduzione al Prompt Engineering 299
23.1 Cos'è un prompt e perché è così importante 301
23.2 Dal prompt semplice al prompt complesso 302
23.3 Il concetto di prompt shaping e prompt chaining 305
23.4 Prompt Engineering vs Prompt Design 308
23.5 Perché il prompt è una nuova forma di programmazione 310
24. Tecniche Base di Prompt Engineering 315
24.1 Strutturare prompt chiari ed efficaci 315
24.2 Ruoli e contesto 316
24.3 Uso di esempi per migliorare la risposta 319
24.4 Prompt iterativi e di raffinamento 321
24.5 Prompt per estrazione, riassunto e riformulazione 324
25. Strategie Avanzate 327
25.1 Chain-of-Thought Prompting 327

25.2 Self-Consistency e Reasoning Avanzato ... 329
25.3 ReAct (Reasoning + Acting) ... 331
25.4 Prompt multilivello e modulari ... 334
25.5 Metaprompting (Prompt che generano prompt) ... 337
26. Prompt per Compiti Specifici ... 339
26.1 Prompt per scrittura creativa e storytelling ... 339
26.2 Prompt per ricerca accademica e generazione di articoli ... 341
26.3 Prompt per programmazione e debugging ... 344
26.4 Prompt per traduzione e localizzazione ... 345
26.5 Prompt per analisi dati e spiegazione di grafici ... 348
27. Tool e Framework per il Prompt Engineering ... 351
27.1 Prompt marketplace e repository condivisi ... 351
27.2 Prompt Engineering con LangChain e LlamaIndex ... 353
27.3 Automazione del Prompt Testing ... 356
27.4 Versionamento e ottimizzazione dei prompt ... 358
28. Prompt Engineering nelle Applicazioni Reali ... 361
28.1 Chatbot e assistenti virtuali ... 361
28.2 Workflow aziendali e automazioni ... 364
28.3 Prompt per marketing e SEO avanzato ... 366
28.4 Prompt per la cybersecurity e test di sicurezza ... 368
28.5 Prompt nelle piattaforme no-code/low-code ... 371
29. Prompt Engineering e Fine-tuning ... 373
29.1 Quando serve addestrare un modello invece di solo promptarlo ... 373
29.2 Few-shot learning e dataset sintetici ... 375
29.3 Ibridazione: prompt + modelli specializzati ... 377
29.4 Lo Style Transfer tramite prompt ... 378
29.5 L'evoluzione verso i Prompt Agents ... 380
30. Errori Comuni e Come Evitarli ... 383
30.1 Prompt troppo vaghi o ambigui ... 383
30.2 Prompt eccessivamente lunghi ... 385
30.3 Mancanza di istruzioni chiare ... 386
30.4 Conflitti di contesto e memory overflow ... 389
30.5 Strategie di debug dei prompt ... 393
31. Etica, Sicurezza e Futuro del Prompt Engineering ... 397
31.1 Prompt injection e attacchi adversarial ... 397
31.2 Prompt che eludono le policy ... 399

31.3 Impatti etici e legali dell'uso improprio 402
31.4 Verso l'AI auto-prompting e autonoma 405
31.5 Prompt Engineering nel futuro degli AGI 407
32. Cento Esempi di Prompt e Risposte per Ambiti Diversi 409
32.1 Esempi di prompt per la progettazione e sviluppo di agenti intelligenti 411
32.2 Esempi di prompt per la gestione di sistemi 413
multiagente (MAS) 413
32.3 Esempi di prompt per manutenzione evolutiva e 413
correttiva 413
32.4 Esempi di prompt per progettazione backend e 414
microservizi AI-native 414
32.5 Esempi di prompt per progettazione frontend 415
interattivi LLM-based 415
32.6 Esempi di prompt per gestione database e modelli dati 416
32.7 Esempi di prompt per Retrieval-Augmented 416
Generation (RAG) 416
32.8 Esempi di prompt per debugging e test automatizzati 417
32.9 Esempi di prompt per sicurezza, etica e 417
compliance 417
32.10 Esempi di prompt per orchestrazione e worlflow 418
AI-based 418
33.Sviluppo dei prompt per la progettazione esecutiva di agenti intelligenti 419
33.1 Progetto esecutivo di un agente autonomo capace di eseguire task di ricerca su fonti web e sintetizzare le informazioni in report markdown 419
33.2 Generazione del codice per un agente AI che gestisce ticket di assistenza clienti con classificazione 425
33.3 Costruzione di un agente conversazionale in Python che si collega a un database SQLite per rispondere a query sui dati 428
33.4 Definizione della struttura di un agente multi-step per l'analisi finanziaria settimanale, con output CSV 430
34. Conclusione Parte III 447
Bibliografia Parte III 453

Parte I: Large Language Models

1. Introduzione

L'Intelligenza Artificiale (IA) nell'ultimo decennio ha vissuto una trasformazione profonda. Da ambito più tecnico-scientifico, ha iniziato ad influenzare sempre più la vita delle persone, espandendo il perimetro della sua operatività. In questo contesto, i Large Language Model (LLM) hanno svolto un ruolo centrale nella storia recente dell'IA, accelerando l'evoluzione del NLP, ma anche sollevando nuove domande e sfide. Ad esempio: che cosa ha determinato la crescita del deep learning rispetto ai modelli statistici? Come è cambiata la natura degli algoritmi e come si sta evolvendo l'architettura degli LLM? In che modo sono migliorate le loro prestazioni in fase di scaling e quali sono le implicazioni di queste scaling laws? Che cosa è l'architettura Transformer e qual è la base matematica del self-attention? Quali sono i vantaggi e gli svantaggi di ogni tecnica di training per un LLM? Quanto impattano accuratezza, costi, robustness e bias dei diversi modelli sull'adozione delle decisioni strategiche? Quali dati sono necessari per avere un buon fine-tuning di un modello generativo? Come si confrontano gli LLM open e closed source in funzione di metriche come accuratezza, robustness, velocità e costi? Cosa bisogna sapere per il compliance all'AI Act europeo e in funzione di quali criteri è opportuno selezionare un modello open source o closed source per un'applicazione pratica?

Questi sono solo alcuni dei temi che saranno affrontati nel libro e che verranno contestualizzati mediante un'analisi della tecnica di LLM: dall'architettura al training e dal fine-tuning al deployment, nonché dall'ottimizzazione hardware/software ad alcune delle piattaforme di sviluppo leader di mercato (Hugging Face e DeepSpeed). La metodologia adottata, basata sulla matematica (self-attention, encoding posizionale, scaling laws), passerà poi all'analisi tecnica del training e del fine-tuning (finetuning completo, LoRA e QLoRA), fino ad arrivare alla RLHF (Reinforcement Learning from Human Feedback). La scelta di un modello LLM sarà effettuata mediante l'analisi di un benchmarking comparativo per diversi modelli open e closed source, attraverso diverse metriche (accuratezza, velocità, costi, robustezza, etc.), le cui sintesi vengono rappresentate tramite radar chart e istogrammi.

La metodologia di questo libro combina la revisione della letteratura, l'analisi empirica e la sintesi, attraverso la consultazione approfondita di libri di testo, manuali, articoli, pubblicazioni scientifiche, blog e forum di esperti e sviluppatori. In aggiunta, saranno

presi in esame casi d'uso reali e codice funzionante in diversi LLM per testare, valutare e confrontare la performance di ciascun modello dal punto di vista quantitativo, per valutare la qualità delle risposte e per individuare punti di forza e debolezze. Saranno implementati metriche quantitative, benchmarking open e istogrammi comparativi, anche dal punto di vista etico, regolatorio e di data curation critica, come parte di un framework analitico multiparametrico e multivariato per l'identificazione delle scelte strategiche e progettuali maggiormente idonee alla delivery del valore aziendale. Particolare importanza sarà data alle implicazioni strategiche che le scelte di scaling, fine-tuning parametrico (LoRA, QLoRA), quantizzazione, containerizzazione e best practice hanno sull'ottimizzazione dei costi e per la gestione dei punti aperti sul bias e sulla sostenibilità.

Per rispondere in modo puntuale e specifico ai seguenti quesiti: che cosa rende gli LLM più potenti dei modelli tradizionali? Come massimizzare l'impatto e la compliance degli LLM in produzione, riducendo al contempo il rischio di bias e i costi? Quali criteri empirici dovrebbero guidare la scelta tra un modello open o closed source? Ogni capitolo del libro sarà dunque composto sia dalla teoria di ogni argomento (architettura, training, fine-tuning, benchmarking, etc.) sia da una pratica (esercizio, ottimizzazione, test, best practice, etc.) atta ad applicare e a utilizzare la teoria affrontata. Ogni capitolo, quindi, avrà lo scopo di fornire agli sviluppatori e ai ricercatori le conoscenze, le linee guida e le risorse pratiche per progettare, valutare e rilasciare LLM.

La struttura del libro sarà così articolata: nei primi capitoli, si introdurranno i LLM, le loro caratteristiche e il loro contesto tecnico, scientifico e di business. Si descriverà il cambiamento tra modelli statistici e deep learning, anche mediante l'analisi delle scaling laws per il growth degli LLM, analizzando i concetti chiave di self-attention, encoding posizionale e scaling. Ci sarà poi spazio all'analisi delle tecniche di training (fine-tuning, LoRA/QLoRA, RLHF) per LLM, ai dati che servono e come bisogna gestirli (data curation). I capitoli centrali saranno dedicati ai framework per sviluppare LLM (Hugging Face, DeepSpeed), all'ottimizzazione hardware e software degli LLM, dalle strategie di deployment ai criteri di benchmarking e alle best practice di automazione e costruzione di applicazioni basate su LLM. La parte conclusiva conterrà la discussione sull'etica, il bias e la sicurezza degli LLM, la compliance al regolamento europeo AI Act e l'analisi delle prospettive future, focalizzata sui modelli multimodali, auto-adattativi e sulle nuove frontiere hardware e software. Le appendici contengono invece aspetti complementari e operativi, come il glossario, la checklist dei modelli open e closed source, le tabelle comparative dei principali LLM, il codice di base per utilizzare ogni LLM.

1.1 Evoluzione dei modelli linguistici

La progressione dai modelli statistici ai LLM costituisce una pietra angolare dell'elaborazione del linguaggio naturale (NLP). I modelli statistici, come la catena di Markov e i modelli n-grammi, utilizzavano sequenze brevi di token per prevedere le probabilità della parola successiva con una finestra di contesto fissa. Tali modelli hanno mostrato un ottimo rendimento per l'epoca, ma si sono presto dimostrati fallaci nella cattura della semantica e delle relazioni a lungo termine (Patil & Gudivada, 2024), rendendo necessari enormi set di dati per fornire un'adeguata risoluzione linguistica in documenti lunghi.

L'avvento delle reti neurali deep learning ha cambiato il paradigma di elaborazione, sostituendo i modelli statistici con un apprendimento di rappresentazioni distribuite per il linguaggio. Questa maggior flessibilità ha consentito una capacità di risoluzione notevolmente più ampia, e il rendimento in problemi di traduzione automatica, question answering, e generazione di testo ha superato le performance precedentemente ottenute con i modelli statistici. Questa rivoluzione del NLP si è potuta sviluppare ulteriormente grazie alla disponibilità di più dati per il training dei modelli e di calcoli ad alte prestazioni (Patil & Gudivada, 2024).

Nel 2017, Vaswani et al. (2017) hanno rilasciato un paper innovativo ("Attention is All You Need"), presentando la prima architettura Transformer e sostituendo l'approccio sequenziale delle RNN con un efficiente calcolo parallelo. Nell'elaborazione NLP, questo modello ha portato grandi miglioramenti nelle prestazioni e ha consentito una forte scalabilità, oltre che di modellare le dipendenze a lungo termine nei dati (Vaswani et al., 2017). L'innovativo meccanismo di self-attention permette al Transformer di gestire tutte le posizioni in una sequenza contemporaneamente e contemporaneamente ad apprendere il significato e la semantica delle posizioni più importanti in essa contenuta.

I vantaggi dell'architettura Transformer sono che questo calcolo parallelo è possibile sia nel training che nell'inferenza. Si ottiene quindi una scalabilità di calcolo che riduce drasticamente il tempo di training. Nei test di traduzione automatica, i modelli Transformer hanno battuto lo stato dell'arte e, pur ottenendo di oltre 2 punti il punteggio BLEU, hanno impiegato un quarto del tempo di training rispetto alle reti LSTM e GRU (Vaswani et al., 2017). L'alta scalabilità, fornita dal Transformer, ha aperto la via ai moderni modelli LLM, e ad architetture sempre più estese e costose dal punto di vista

computazionale.

Per capire come, all'aumentare della grandezza di modelli e set di dati, le prestazioni possono migliorare, Kaplan et al. (2020) hanno individuato empiricamente le cosiddette laws of scaling.

È stato infatti scoperto che la perdita nei modelli diminuisce in funzione della grandezza dei parametri, dei dati e dei calcoli tramite una legge di potenza, che però presenta rendimenti decrescenti, ed è quindi da tenere in conto un certo limite oltre il quale incrementare ancora la grandezza dei modelli non implica più un sensibile aumento delle prestazioni. Questa diminuzione della perdita, grazie alla grandezza dei modelli, comporta la necessità di un numero sempre inferiore di esempi e step di ottimizzazione per un raggiungimento di accuratezza di un dato livello. Se paragonati a modelli più piccoli, i modelli di grande scala si dimostrano più efficaci per task in few-shot learning o in zero-shot learning. Questo perché riescono a raggiungere una generalizzazione predittiva accettabile anche se vengono utilizzati un numero relativamente basso di esempi o step di training. Aumentare l'accuratezza di questo tipo di modelli non ne diminuisce la velocità in fase di inferenza; la maggior parte del tempo di calcolo è spesa nel training. Un'altra interessante proprietà è data dalla generalizzazione predittiva, che mostra come per ottenere le migliori prestazioni si possa investire il budget computazionale in modelli molto grandi, fermando il training prima della convergenza (Kaplan et al., 2020).

Queste scoperte e laws of scaling ci mostrano il potenziale enorme che queste architetture ci permettono di raggiungere. Allo stesso tempo, però, sollevano alcuni interrogativi sulla loro efficacia e sulla scalabilità dei miglioramenti ottenuti solo aumentando l'estensione del modello. Il limite per l'utilità di un ulteriore aumento della grandezza è puramente economico, ma va comunque considerato anche l'aspetto delle ripercussioni ambientali causate da un incremento di consumi di energia elettrica (Kaplan et al., 2020). Ulteriori strategie architettoniche si sono rivelate efficaci nell'aumento delle prestazioni di questi modelli. Self-attention multi-head e positional encoding consentono la modellazione di documenti di lunghezza arbitraria (Han et al., 2021).

Self-attention multi-head permette una migliore e più flessibile modellazione linguistica pesando differenti relazioni tra i token e consentendo al modello di generalizzare attraverso lingue distanti e domini. L'aggiunta dell'encodifica posizionale consente di acquisire e modellare la distanza relativa, rendendo il contesto robusto e migliorando il rendimento nei task che richiedono dipendenze a lungo termine per una corretta comprensione dei dati. Un recente progresso architettonico, i Transformer in

Transformer (TNT), ha dimostrato sostanziali miglioramenti nei risultati di accuratezza. Se modelli simili vengono adottati in questo dominio, anche in esso potrebbe vedersi un simile miglioramento (Han et al., 2021).

Grazie ai grandi passi avanti tecnologici nell'elaborazione del linguaggio naturale, i modelli LLM hanno avuto un forte impatto sull'intera società, aumentando l'innovazione e la conoscenza in numerosi domini come sanità, legge e finanza. Allo stesso tempo, sono emersi due problemi inerenti alla scalabilità dell'elaborazione, alla velocità e al tempo di calcolo che ha richiesto tale avanzamento. I sistemi NLP di questo tipo richiedono alti consumi di energia per addestrare i modelli, incrementando la spesa totale e generando costi imprevisti (Patil & Gudivada, 2024). Inoltre, la facile disponibilità di dati e modelli open source, associata a un più ampio accesso a sistemi di sviluppo software di alto livello, come il framework Hugging Face, ha fornito ai creatori un grande aiuto nell'implementazione di LLM, permettendo un rapido aumento della loro applicabilità. Ciò consente a un più ampio numero di sviluppatori, e localizzati in aree anche più disparate nel mondo, di utilizzarli e ad aggiungere valore ai propri sistemi. Sebbene la maggiore scalabilità di calcolo abbia permesso l'esplosione dello sviluppo di LLM, il loro consumo energetico non è sostenibile dal punto di vista ambientale e richiede nuove politiche di governance e di etica che consentano uno sviluppo responsabile di questi modelli. Il rilascio di modelli LLM più grandi, ma con elevata richiesta di calcolo, richiede la definizione di protocolli governativi al fine di consentire la creazione di politiche etiche sullo sviluppo (Patil & Gudivada, 2024).

In conclusione, possiamo affermare che la progressione dai modelli statistici ai LLM ha portato a una radicale evoluzione nel dominio del NLP.

1.2 Dalla statistica al deep learning

Il passaggio dai modelli statistici classici a quelli basati sul deep learning ha rappresentato una trasformazione importante nella modellazione del linguaggio naturale. I modelli statistici si basavano su un numero limitato di token per calcolare la probabilità del token successivo. Le catene di Markov e gli n-grammi, nonostante abbiano rappresentato un passo in avanti nella modellazione del linguaggio naturale, risultavano limitati alla memoria locale. Questo difetto rendeva le previsioni del token successivo dipendenti dal token (o token) precedente, ma inesplicabile dal punto di vista del contesto. Le limitazioni, quindi, portavano a un aumento delle dimensioni e ad alta complessità computazionale dei modelli, che, con l'aumento del vocabolario,

diventavano rapidamente intrattabili. Questo difetto richiedeva enormi quantità di dati, dei quali però i modelli statistici ne sfruttavano solo una minima parte, non offrendo quindi prestazioni soddisfacenti in task complessi (Patil & Gudivada, 2024).

Le reti neurali profonde hanno introdotto un'alternativa di più alto livello, rendendo dinamicamente variabili i numeri in rappresentazione vettoriale statica dei modelli statistici classici. Infatti, grazie alla tecnica degli embedding distribuiti, si è dimostrato possibile rappresentare informazioni semantiche e sintattiche e catturare relazioni complesse fra le parole. Questo modello di rappresentazione si è rivelato efficace e più espressivo, portando all'introduzione di tecniche più avanzate che hanno contribuito alla generalizzazione del campo. I primi approcci con le reti neurali ricorrenti (RNN) e varianti più sofisticate come le LSTM (Long Short-Term Memory) sono riuscite a migliorare significativamente le capacità di modellazione per i dati sequenziali. Purtroppo, in entrambi i casi, non sono state del tutto risolte le limitazioni dovute al vanishing gradient e alla difficoltà di parallelizzazione, rimanendo comunque meno adatte ai dati sequenziali rispetto ai Transformer (Patil & Gudivada, 2024).

L'architettura del Transformer ha stabilito un nuovo stato dell'arte nei modelli del linguaggio naturale. Rispetto alle RNN, presenta diverse caratteristiche innovative, in particolare il meccanismo di self-attention, che permette al modello di processare tutte le posizioni della sequenza in simultanea. Questo si traduce in un incremento di efficienza e parallelizzazione in fase di training, migliorando le performance nei task complessi. La self-attention rende la modellazione delle dipendenze di lungo raggio più accurata, perché permette al modello di pesare differenzialmente l'importanza di ogni coppia di token. In questo modo, il modello capisce e codifica in modo più appropriato le relazioni fra tutti i token presenti nei dati testuali, rappresentando il primo passo per una comprensione semantica del testo (Manning & Hewitt, 2023). L'introduzione dell'encoding posizionale elimina, inoltre, l'obbligo di informazioni sequenziali di ogni input, permettendo al modello di capire l'ordine degli input processati anche in parallelo. Le performance ottenute nei task di machine translation e question answering hanno dimostrato l'efficacia dell'architettura Transformer, imponendola come lo standard nel campo (Kaplan et al., 2020).

Le scaling laws si sono occupate di quantificare la dipendenza delle performance degli LLM dalle dimensioni del modello, dai dati di training e dalla potenza computazionale utilizzata per l'addestramento. Le dimensioni del modello (in termini di parametri del network) si sono rivelate un parametro cruciale: modelli più grandi generalmente raggiungono performance migliori rispetto a quelli più piccoli. Le scaling laws sono anche predittive: modelli più grandi imparano più velocemente di quelli più piccoli,

richiedendo una quantità inferiore di dati o passaggi di training per raggiungere un determinato livello di performance. Questa predittività è fondamentale per ottimizzare le dimensioni del modello in base alle risorse computazionali a disposizione. Tuttavia, i benefici della scalabilità in realtà presentano rendimenti decrescenti oltre una certa soglia, e più in generale dipendono da quanto è pulito il training dataset. Infatti, seppure l'aumento delle dimensioni dei dataset abbia garantito prestazioni migliori rispetto al semplice ridimensionamento dei modelli, in alcuni casi i vantaggi risultano limitati dalla presenza di "rumore" nei dati. In questa direzione, la data curation è diventata sempre più importante (Kaplan et al., 2020).

A causa del fatto che il deep learning ha la tendenza a "ricordare" i pattern presenti nei dati di training, le problematiche dell'etica in materia di IA hanno assunto grande importanza, soprattutto per quanto riguarda i bias, e sono state risolte mediante le attività di validazione dati e il monitoraggio delle prestazioni dei modelli. La maggiore disponibilità degli open source framework (Hugging Face) ha facilitato la partecipazione al movimento open source in questo campo, rendendo sempre più accessibile l'esperienza di building di reti neurali artificiali. Questo passaggio dalla fase della teorizzazione al test sul campo della maggior parte degli sviluppatori ha accelerato i progressi e contribuito all'adozione di standard etici che siano adatti agli LLM (Floridi, 2022).

Il cambio di paradigma verso il deep learning non ha tuttavia portato solo progressi. Lo sviluppo degli LLM è sostenibile per l'ecosistema? La domanda nasce spontaneamente se pensiamo alle dimensioni dei modelli e, quindi, dei dataset, la complessità delle architetture e la tecnologia hardware utilizzata, come GPU e TPU, che si traducono in un consumo di energia importante, come risorsa energetica e inquinante, e un forte impatto ambientale in termini di anidride carbonica emessa durante la fase di addestramento, pari a circa cinque automobili in ciclo di vita (Patil & Gudivada, 2024). Nonostante l'addestramento degli LLM si svolga a basso regime energetico e in data center ecologici e le emissioni siano ripagate dalla migliore efficacia e dall'incremento del workflow nell'elaborazione del linguaggio naturale, i benefici sono inferiori al rapporto costi e benefici. Lo sviluppo di tecniche per addestrare modelli complessi e di grandi dimensioni (quantizzazione, pruning e architetture più performanti), la regolamentazione sull'utilizzo dei dati (rispetto della privacy, trasparenza sull'origine e copyright) e l'utilizzo degli open source dataset permetteranno in futuro una modellazione sostenibile, efficace ed etica (Floridi, 2022).

1.3 Il Transformer e il paper 'Attention is All You Need'

L'articolo "Attention is All You Need" rappresenta una milestone nel campo dell'elaborazione del linguaggio naturale (NLP), in quanto propone il Transformer come alternativa alle RNN e alle CNN. La sua caratteristica innovativa risiede nell'eliminazione della dipendenza dall'ordine, elaborando contemporaneamente tutte le posizioni della sequenza con l'attenzione, il che gli consente di trattare meglio le relazioni di lungo raggio (Vaswani et al., 2017; Manning & Hewitt, 2023).

L'architettura Transformer è il cuore di questo modello, il cui principale punto di forza risiede nel fatto che si basa solo sul meccanismo dell'attenzione e non utilizza più l'architettura sequenziale delle RNN e delle LSTM. In questo modo è possibile effettuare una parallelizzazione su larga scala durante l'addestramento, eliminando i problemi del gradiente scomparso e della necessità di operare in contesti ampi (Manning & Hewitt, 2023).

È stato ben presto scelto come riferimento nella pratica del NLP poiché dimostra buone performance e, allo stesso tempo, è stato testato su vari domini e compiti. La capacità di apprendimento è migliore dei competitor con una maggiore flessibilità (Patil & Gudivada, 2024).

L'architettura introdotta da Vaswani et al. è altamente scalabile in termini di profondità (numero di layer) e larghezza (dimensione delle matrici dei pesi) e permette di sviluppare modelli task-agnostic e multilingua. Ciò lo rende di fondamentale importanza anche in un futuro per i modelli multimodali e dell'AGI, poiché la capacità di generalizzare è ottimale (Manning & Hewitt, 2023).

Il paper innovativo risiede nel sostituire l'architettura sequenziale con una matrice di attenzione: esso permette così l'apprendimento delle relazioni di ogni token con il resto del dataset e il processo diventa facilmente parallelizzabile. Perfezionando il meccanismo dell'attenzione, i parametri per i pesi degli archi si imparano autonomamente e non c'è bisogno di operazioni di embedding e di decoding in ordine sequenziale (Vaswani et al., 2017).

La self-attention multi-head utilizza un meccanismo in cui le query, i key e i value sono sottoposti a una trasformazione lineare proiettandoli su diverse dimensioni, così da apprendere relazioni fra gli elementi in posizioni diverse. La divisione dell'attenzione in head multipli permette anche di valutare diverse interpretazioni delle relazioni fra i vari token (Manning & Hewitt, 2023).

Con l'abbandono dell'elaborazione sequenziale è possibile raggiungere una completa parallelizzazione degli addestramenti. In questo modo è diminuito notevolmente il

tempo di convergenza ed è stato possibile scalare l'LLM ad un livello che, in precedenza, non era possibile raggiungere. I Transformer hanno dominato il task della traduzione automatica, dimostrandosi più efficienti degli state-of-the-art di quel momento (Vaswani et al., 2017).

Le sperimentazioni del paper confermano l'efficacia del Transformer. Ad esempio, il modello base è stato addestrato in meno di dodici ore con otto GPU P100 (Vaswani et al., 2017).

La self-attention multi-head costituisce la base su cui costruire dipendenze complesse, multi-turno e cross-dominio, come ad esempio modelli multimodali e agenti cognitivi (Patil & Gudivada, 2024).

Grazie alla sua capacità di parallelizzazione, questo meccanismo è implementabile su hardware che usano appieno le potenzialità delle moderne GPU e TPU. Le efficienze ottenute sono superiori rispetto ai sistemi tradizionali, e le GPU sono ampiamente disponibili e adatte a questi addestramenti grazie alla potenza di calcolo che possiedono, e sono adatte anche ad ottimizzare la memoria con un uso ridotto di CPU (Manning & Hewitt, 2023).

Più si cerca l'efficienza, più aumenta il rischio di quadratic complexity, a causa dell'incremento della lunghezza della sequenza, ed è necessario pensare ad ottimizzazioni come tecniche di attenzione efficienti e compressione delle matrici dei pesi (Wang et al., 2020).

Questo modello favorisce lo sviluppo dei modelli generalisti, perché permette di sfruttare in modo più semplice il self-supervised learning e il transfer learning. Questo ha reso possibile il pre-training su dataset larghi, una pratica standard su cui si baserà anche l'evoluzione futura degli LLM (Patil & Gudivada, 2024).

L'attenzione è ottenuta per mezzo di prodotti scalari di matrici pesi (query, key, value) con opportune normalizzazioni e con la funzione softmax (Vaswani et al., 2017).

Il vantaggio di tale meccanismo consiste nella trasformazione dell'apprendimento di una sequenza di dati, eliminando così la dipendenza dagli stati precedenti, in un calcolo puramente matriciale, che offre anche maggiori vantaggi e potenzialità nell'implementazione su calcolatori paralleli. Dalla dimensione temporale si passa a quella spaziale e questo è in linea con l'hardware moderno (Arora & Barak, 2007).

In quanto a complessità, l'architettura Transformer ha scambiato la profondità con la memoria. Infatti, questo sistema, elaborando in parallelo tutte le posizioni dell'input, presenta profondità inferiori rispetto ai sistemi precedenti, il che favorisce l'apprendimento semantico anche su testi complessi (Wang et al., 2020).

1.4 Perché i modelli sono 'large' e come scalano

Le motivazioni per cui i modelli di linguaggio naturale sono definiti 'large' e il loro approccio di scaling rappresenta una discussione essenziale nella comprensione degli LLM. Kaplan et al. (2020) hanno empiricamente dimostrato le scaling laws che definiscono una correlazione tra la dimensione del modello, la dimensione del dataset e la potenza computazionale, mostrando un miglioramento predittivo sistematico e riproducibile. La crescita della scala conduce a un'efficienza di campionamento perché necessita di meno passi di ottimizzazione con meno dati, ottenendo performance simili a quelle dei modelli più piccoli. I risultati hanno avuto un forte impatto sia nel mondo accademico sia in laboratori e aziende, contribuendo a rendere lo scaling una strategia di alto interesse ed efficacia per raggiungere diversi benchmark.

Le scaling laws stabiliscono una relazione matematica per cui la perdita decresce come legge di potenza con i parametri, i dati e la potenza computazionale, ma oltre un certo punto le performance ottenute non riescono a superare i rendimenti decrescenti (Kaplan et al., 2020). Per questo, l'aumento dei parametri e del dataset può rivelarsi eccessivo per il miglioramento predittivo, comportando un costo maggiore. I rendimenti decrescenti sono legati al fatto che l'errore può essere solo abbassato fin quando si converge a zero e più parametri e dati non implicano performance più elevate. In questo scenario, le scaling laws possono essere utilizzate come linee guida per trovare rendimenti decrescenti. In definitiva, l'espansione della scala non è una soluzione per tutti i problemi di architettura e metodologia, e si sono sviluppate tecniche che hanno massimizzato l'efficienza senza aggiungere semplicemente parametri.

Come predetto da Alabdulmohsin et al. (2022), l'eccesso di perdita decrementa con una legge di potenza decrescente con il numero di parametri e la dimensione del dataset, rendendo lo scaling un'area di interesse con i rendimenti decrescenti. Sebbene le scaling laws stabiliscano la relazione matematica tra le performance e la scala, questo scenario teorico non ha rappresentato una base sufficiente per l'effettivo scaling dei modelli, stimolando la ricerca di compressione dei modelli tramite il pruning o la quantizzazione, o metodologie di fine-tuning più efficaci che utilizzino solo un piccolo set di dati senza dover accrescere ulteriormente la scala del modello.

Gli LLM più estesi hanno cominciato ad emergere solo recentemente, in parte per l'accresciuta disponibilità di potenza computazionale e large dataset, il cui incremento negli ultimi due anni ha guidato un'accelerazione nello sviluppo di modelli sempre più estesi e di parametri più numerosi (decine e centinaia di miliardi) (Patil & Gudivada,