



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Algoritmi di Intelligenza artificiale per lo sviluppo di un sistema di analisi automatica di fenomeni ambientali

RELATORE

Prof. Fabio Palomba

Ing. Daniele Sofia

Università degli Studi di Salerno

CANDIDATO

Francesco Peluso

Matricola: 0512107194

Anno Accademico 2021-2022

Questa tesi è stata realizzata nel

sesa^{lab}
SOFTWARE ENGINEERING
SALERNO

" - SAM DRAKE: Pensavo che... trovato il tesoro di Avery mi sarei sentito... appagato?

Invece mi... mi ha lasciato uno strano senso di...

- NATHAN DRAKE: Vuoto.

- SAM DRAKE: Esatto.

- NATHAN DRAKE: Beh, vedi, per quanto un'avventura sembri esaltante, quando finisce ti lascia sempre quella sensazione. Prima o poi devi decidere a cosa tieni davvero e... cosa lasciarti alle spalle."

Dialogo tratto da "Uncharted 4: Fine Di Un Ladro", 2016, sviluppato da Naughty Dog.

Sommario

La tesi presentata offre un'iniziale panoramica generale del mondo del Natural Language Processing, delle sue branche e principali attività. Di conseguenza si mostra l'utilizzo di tecniche di Deep Learning per risolvere problematiche relative all'NLP, le quali in poco tempo si sono rapidamente evolute, soprattutto per i modelli di linguaggio. Tutto questo però, per introdurre l'argomento centrale dell'elaborato, ovvero l'utilizzo di GPT-2 per la modellazione e la generazione di report ambientali, strettamente relati a dati reali. Vengono così presentate una porzione delle attività proposte dalla text generation, che hanno permesso di raggiungere lo stato dell'arte. Tra le varie tecniche, verrà analizzata principalmente quella dei Transformer, presentando le varie tipologie presenti, utile per introdursi verso l'utilizzo di GPT-2. Attraverso GPT-2, si è realizzati un sistema a due layer, per la generazione di report ambientali, dove ogni report non sia in contraddizione con la realtà. Infine tale sistema si è dimostrato soddisfacente per uno sviluppo iniziale, che farà da base per i miglioramenti futuri.

Indice

Indice	i
Elenco delle Figure	iii
Elenco delle Tabelle	v
1 Introduzione	1
1.1 Sense Square e la qualità dell'aria	1
1.2 Motivazioni e Obiettivi	4
1.3 Risultati ottenuti	4
1.4 Struttura della tesi	5
2 Background e Stato dell'arte	6
2.1 Natural Language Processing	6
2.1.1 Natural Language Understanding	7
2.1.2 Natural Language Generation	8
2.2 Deep Learning	9
2.2.1 Reti Neurali	12
2.2.2 Vantaggi e difficoltà dell'utilizzo del Deep Learning	16
2.3 Modelli di linguaggio per la generazione del testo	17
2.3.1 Modello linguistico basato su n-grams	18

2.3.2	Modelli linguistici basati su Deep Learning	19
2.4	GPT-2	25
2.4.1	Architettura e funzionamento	27
3	Generazione di report ambientali tramite l'uso di intelligenza artificiale	35
3.1	Specifiche tecniche e librerie utilizzate	37
3.2	Creazione dataset con dati strutturati	39
3.3	Creazione dataset con dati non strutturati	43
3.4	Addestramento	48
3.5	Funzione di generazione	49
3.6	Pre-processing dell'input	50
3.7	Messa in produzione	56
4	Valutazione	59
4.1	Script di test	59
4.2	Risultati	65
5	Conclusioni e sviluppi futuri	66
	Bibliografia	68

Elenco delle figure

1.1	Piattaforma per il monitoraggio della qualità dell'aria. [1]	2
1.2	Livello di inquinamento per la regione Campania. [1]	2
1.3	Monitoraggio giornaliero degli inquinanti. [1]	3
2.1	Suddivisione dell'NLP. [2]	7
2.2	Branche dell'AI. [3]	11
2.3	Neurone. [4]	12
2.4	Neurone all'interno di una rete neurale artificiale. [5]	13
2.5	Rete neurale artificiale. [6]	13
2.6	Funzione di attivazione. [7]	14
2.7	Recurrent Neural Network (RNN). [8]	22
2.8	Long Short-Term Memory (LSTM). [9]	23
2.9	Rappresentazione di una rete Transformer. [10]	24
2.10	Modelli GPT, BERT. [11]	25
2.11	Varianti del modello di GPT-2. [12]	26
2.12	Architettura transformer. [12]	27
2.13	Modelli GPT, BERT, TRANSFORMER XL. [12]	27
2.14	Blocco Encoder. [12]	28
2.15	Blocco Decoder. [12]	28
2.16	Generazione token(1). [12]	29

2.17	Generazione token(2). [12]	30
2.18	Generazione token(3). [12]	30
2.19	Differenze tra Self-Attention e Masked Self-Attention. [12]	31
2.20	Architettura GPT-2. [12]	31
3.1	Pipeline eseguita.	36
3.2	Visualizzazione parziale del dataset finale con dati strutturati.	41
3.3	Visualizzazione parziale dei dati raccolti.	51
3.4	Visualizzazione parziale del file delle frasi generate.	58
4.1	Province-regioni scelte casualmente.	60

Elenco delle tabelle

3.1	Differenze delle versioni fornite da GPT-2-SIMPLE [13].	37
-----	---	----

CAPITOLO 1

Introduzione

1.1 Sense Square e la qualità dell'aria

L'aria è un miscuglio di sostanze aeriformi che costituisce l'atmosfera terrestre. È indispensabile per la vita della maggior parte degli organismi viventi, in particolare per l'uomo poiché necessaria per la respirazione. Oltre all'ossigeno necessario, ad ogni respiro, inaliamo anche piccole quantità di gas potenzialmente dannosi. Questi componenti influenzano direttamente la nostra salute e dunque, con il passare del tempo, l'uomo ha iniziato a preoccuparsi per la qualità dell'aria. Quando si parla di qualità dell'aria ci si riferisce alla presenza di inquinanti nell'aria ambiente. La presenza di quantitativi significativi di inquinanti, rispetto ai valori limite individuati dalla legge, pregiudica la qualità dell'aria. Esistono diverse aziende che si occupano del monitoraggio della qualità dell'aria, in particolare Sense Square.

Sense Square Srl è una start-up innovativa specializzata nella creazione di reti tecnologiche per la raccolta e aggregazione di dati territoriali georeferenziati. Nello specifico, l'azienda ha sviluppato un sistema automatico di tracciamento delle sorgenti di inquinamento. Il fine della start-up è quello di rendere il monitoraggio della qualità dell'aria un diritto di tutti i cittadini (attraverso una piattaforma)(Fig 1.1), così

da renderli consapevoli dell'aria che respirano, permettendogli, così, di assumere comportamenti ed intraprendere azioni per limitare gli impatti negativi sulla propria salute.

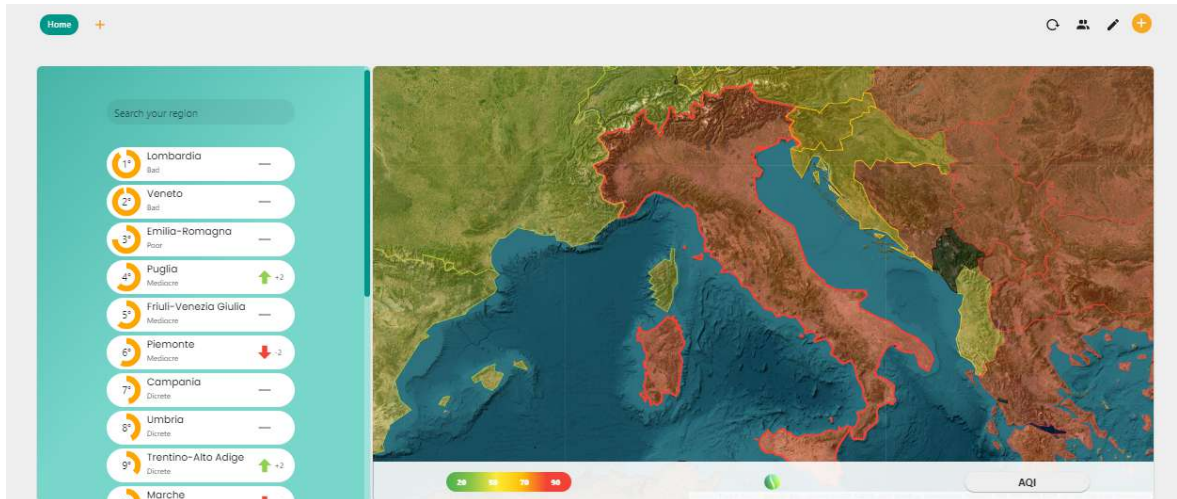


Figura 1.1: Piattaforma per il monitoraggio della qualità dell'aria. [1]

Con i dati a disposizione, l'azienda Sense Square può analizzare e monitorare la qualità dell'aria a diversi livelli di dettaglio, partendo da intere nazioni fino ai singoli km2 (Fig.1.2).

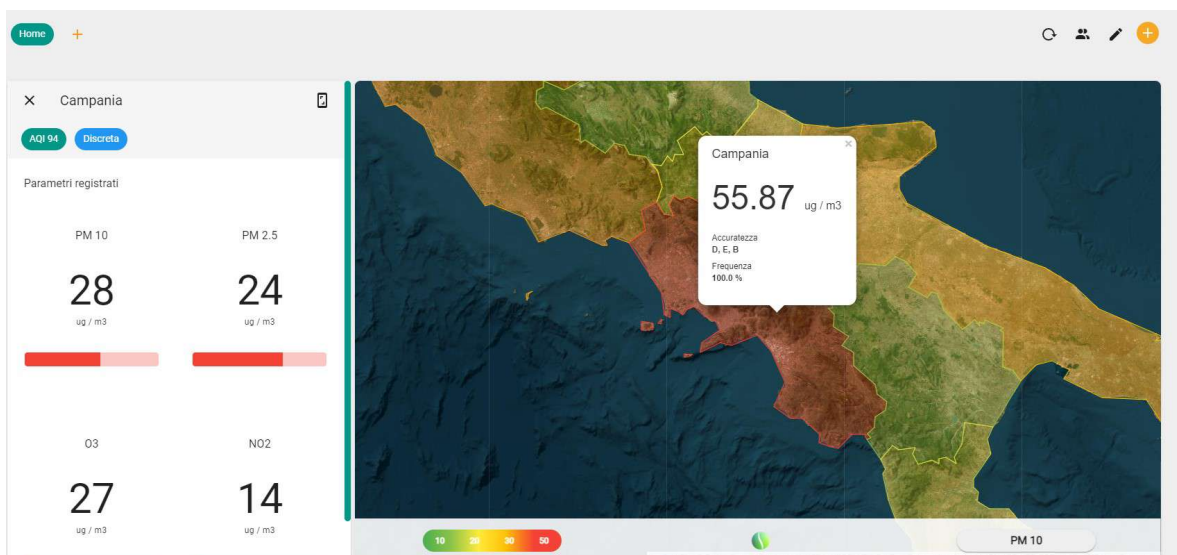


Figura 1.2: Livello di inquinamento per la regione Campania. [1]

Grazie a tale piattaforma, Sense Square riesce a monitorare e visualizzare le percentuali di diffusione dei principali inquinanti dell'aria:

- AQI
- PM1
- PM2,5
- PM10
- VOC
- NO2
- CO
- O3
- SO2
- H2S

Tali percentuali possono essere distribuite in base alla media giornaliera, alla media oraria e alla media storica. Questa distribuzione consente di poter avere un controllo completo, anche grazie alla possibilità di poter scegliere tra diversi tipi di grafici (come, ad esempio, il grafico a barre, linea...) (Fig.1.3).

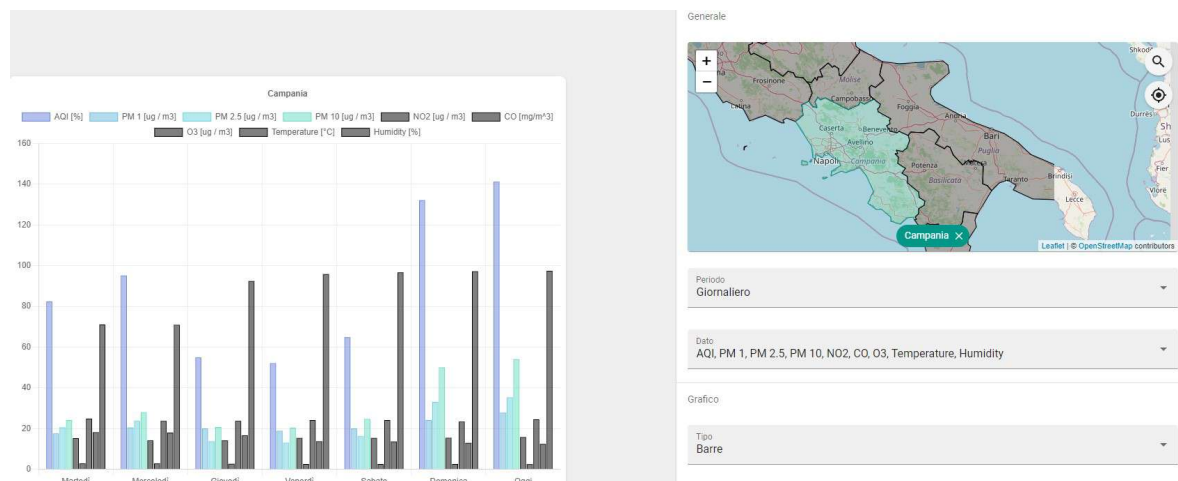


Figura 1.3: Monitoraggio giornaliero degli inquinanti. [1]

1.2 Motivazioni e Obiettivi

Per migliorare ulteriormente il monitoraggio della qualità dell'aria, in modo da poter rendere i cittadini sempre più consapevoli delle condizioni ambientali, sulla piattaforma di Sense Square, è nata la necessità di sviluppare un sistema del tutto automatico ed intelligente. Il sistema proposto, deve essere in grado di generare dei piccoli report ambientali, che descrivono in maniera accurata lo stato della qualità dell'aria ed eventi storici di una determinata provincia o regione, partendo da dati reali o proprietari. Quindi le motivazioni della stesura del seguente elaborato, sono quelle di realizzare un sistema di Deep Learning che soddisfi gli obiettivi preposti da Sense Square.

1.3 Risultati ottenuti

Attraverso l'uso di GPT-2, si ottiene un quadro completo sulle moderne tecniche di Intelligenza Artificiale, che non solo vengono applicate su diversi contesti, ma sono tutt'oggi in continua evoluzione. Infatti, si sa che GPT-2 non è l'ultima versione della serie GPT, ma attualmente, risulta essere l'ultima della serie che sia più aggiornata (rispetto ai suoi predecessori) e open source, che fa al caso di una start-up. I risultati ottenuti non vogliono aprire nuovi orizzonti a differenti approcci sullo studio, ma fanno più da panoramica generale a tecniche già esistenti, che a loro volta, fanno da base ad un sistema che dovrà essere più complesso in futuro. Infine, è stato realizzato un sistema a due layer, tra cui racchiude:

- Un layer dedicato al modello per la generazione di report ambientali, che fornisce indicazioni veritiere, ovvero che non siano in contraddizione con la realtà.
- Un layer dedicato puramente al pre-processing dell'input, necessario per avere un output conforme con la realtà, da parte del modello.

Tale modello è stato valutato "a mano", a causa della relazione stretta con la realtà, al fine di ricavarne le prestazioni e possibili miglioramenti., che contribuiranno con il miglioramento del modello nel futuro, per ottenere dei report sempre più dinamici.

1.4 Struttura della tesi

Il seguente elaborato è strutturato in vari capitoli, ognuno con lo scopo di presentare e descrivere in modo approfondito diversi aspetti della ricerca effettuata e del modello sviluppato. Nel Capitolo 1 sono elencati gli obiettivi e le motivazioni relative all'esigenza di dover costruire un modello per un miglioramento del monitoraggio dei dati. Nel Capitolo 2 è analizzato il Background e lo Stato dell'Arte, ossia ci si concentra sul NLP in generale da un punto di vista puramente teorico, approfondendo una particolare branca di NLP, denominata NLG. Si parte dai primi modelli di generazione del testo, arrivando infine alle ultime architetture di rete usate nell'ambito del Natural Language Generation, nello specifico quelle della serie GPT. Nel Capitolo 3 vengono presentati i vari layer del sistema finale. Nel primo layer si parla dell'analisi effettuata sui dati provinciali, per il pre-processing dell'input. Nel secondo layer si definisce la pipeline di Machine Learning impiegata per l'addestramento del modello per la generazione dei report ambientali, presentando le tecnologie utilizzate, le tecniche di Pre-Processing applicate ai dati e il possibile utilizzo del modello. Nel Capitolo 4 saranno descritti tutti i risultati che sono stati ottenuti, cercando di valutarli in maniera empirica. Infine, nel Capitolo 5 saranno presentate le conclusioni ed eventuali sviluppi per lavori futuri.

Background e Stato dell'arte

2.1 Natural Language Processing

Il Natural Language Processing è una branca della computer science e della linguistica. Si occupa di fornire ai sistemi intelligenti la possibilità di leggere e capire il linguaggio utilizzato dagli esseri umani (come, ad esempio, un testo scritto o una conversazione orale) in una maniera che sia produttiva ed efficiente [14]. La difficoltà principale di questo processo è l'intrinseca ambiguità che caratterizza i linguaggi naturali, per questo motivo le soluzioni richiedono un'estesa conoscenza e una notevole abilità nel manipolarla. Infatti, si deve tener conto della struttura, della grammatica e di un lessico adeguato al contesto. Per tener conto di ciò, si utilizza una fase di pre-processing, di fatti, nel Natural Language Processing, esistono due fasi principali: data pre-processing e lo sviluppo di algoritmi. La fase di data pre-processing consiste nel preparare e pulire i dati testuali per renderli analizzabili dalla macchina, in modo tale che essi siano in una forma lavorabile [15]. Ci sono molti modi con cui può essere fatto, tra i quali: [16]

- Tokenization: che si occupa di suddividere il testo in unità minime di analisi (chiamate token). La suddivisione in token rende possibili gli ulteriori step di pre-processing

che, altrimenti, risulterebbero complessi o inefficaci. Esistono due tipi di Tokenization: La sentence tokenization, che consiste di avere come unità minima di analisi un'intera frase e la word tokenization, che suddivide l'intero testo in parole.

- La rimozione dei “noisy data”, ovvero eliminare i dati rumorosi (ad esempio un link...), in modo tale che vengano lasciate nel testo solo i dati rilevanti.
- Lemmatization e stemming, per ridurre le parole alla loro radice.

Infine, l'intera branca dell'NLP si suddivide in due categorie, NLG e NLU. [14]

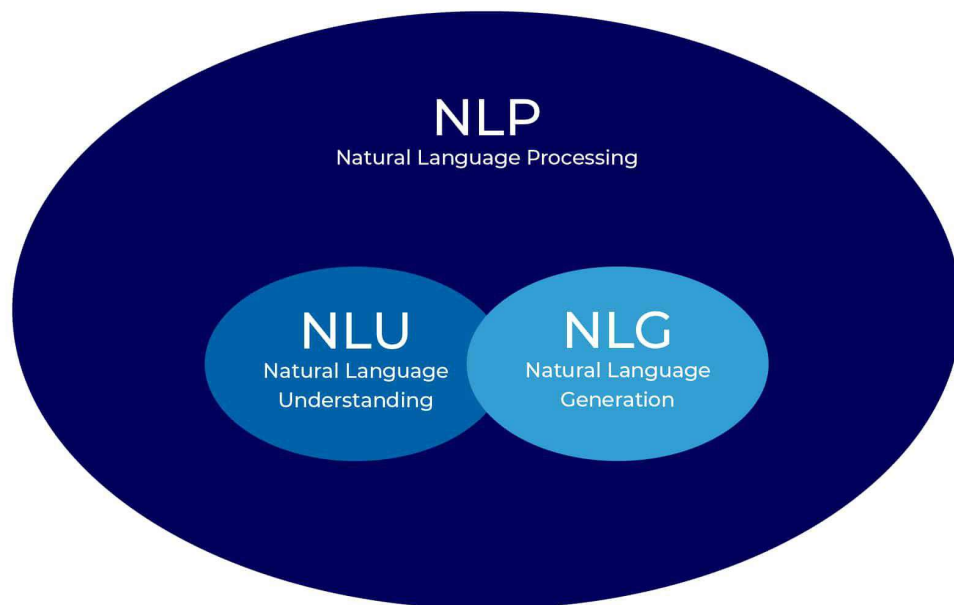


Figura 2.1: Suddivisione dell'NLP. [2]

2.1.1 Natural Language Understanding

Lo scopo dell'NLU (Natural Language Understanding) riguarda la capacità delle macchine di comprendere e utilizzare l'analisi sintattica e semantica per determinare il significato di un testo. Esso può essere applicato ad una serie di processi, come la classificazione dei testi, la raccolta di informazioni e l'analisi dei contenuti. Il Natural Language Understanding è un campo in evoluzione e mutevole ed è considerato uno dei problemi più difficili dell'IA. Sono in fase di sviluppo varie tecniche e strumenti per fornire alle macchine una comprensione del linguaggio umano [17].

Le varie tecniche includono: [17]

- Il riconoscimento di entità nominate: è il processo che opera distinguendo concetti e riferimenti fondamentali in un corpo di testo, identificando entità nominate e inserendole in categorie come luoghi, date, organizzazioni, persone, lavori, ecc.
- La disambiguazione del senso della parola: è il processo di determinazione del significato, o senso, di una parola in base al contesto in cui la parola appare.

Alcuni esempi comuni di NLU includono: ragionamento automatico, instradamento automatico dei ticket, traduzione automatica e risposta alle domande. [17]

2.1.2 Natural Language Generation

Per Natural Language Generation(NLG) intendiamo l'insieme di tecniche e algoritmi per la generazione automatica di informazioni scritte in linguaggio naturale, che è quello della lingua parlata. Nel caso applicativo di chatbot e assistenti vocali, esso fa parte dell'ultima fase del processo in cui viene compreso un input in linguaggio naturale, viene trasformato in un insieme di dati strutturati, ne viene elaborata una risposta o un'azione, infine l'output viene trasformato in un linguaggio simile a quello di input, che spesso è la lingua parlata e a volte tradotto in un'altra lingua per i sistemi più evoluti [18]. Tra i vantaggi nell'uso di NLG possiamo trovare: [18]

- Risparmio di tempo
- Creazione di contenuti in modo automatico, ovvero disporre di sistemi che producono contenuti testuali senza l'intervento umano.
- Scalabilità in tempo reale.

Tra le possibili applicazioni si può pensare a contesti in cui in sostituzione di un report o di una comunicazione schematica, si può produrre attraverso un documento nel linguaggio naturale più familiare e comprensibile; messaggi informativi nel settore ambientale, nel settore trasporti, nelle diverse branche della medicina, nella fruizione di contenuti web. Infine la necessità e la creatività determinano i limiti applicativi.

2.2 Deep Learning

Il deep learning è un metodo di intelligenza artificiale (IA) che insegna ai computer a elaborare i dati in un modo che si ispira al cervello umano. I modelli di deep learning sono in grado di riconoscere pattern complessi in immagini, testo, suoni e altri dati per produrre informazioni e previsioni accurate. Si possono utilizzare metodi di deep learning per automatizzare le attività, che in genere richiedono l'intelligenza umana, come la descrizione di immagini o la generazione di testo. Il deep learning è un campo specifico dell'apprendimento automatico (machine learning), che include modelli statistici e predittivi. Attraverso il deep learning si possono creare dei modelli di analisi predittiva, che apprendono da dati correnti e passati per predire attività, comportamenti e tendenze future. Gli algoritmi di deep learning sono emersi nel tentativo di rendere più efficienti le tecniche di machine learning tradizionali [19]. I metodi tradizionali di machine learning richiedono un notevole sforzo umano per addestrare il software. Ad esempio, nel riconoscimento delle immagini di animali, è necessario eseguire le seguenti operazioni: [19]

- Etichettare manualmente centinaia di migliaia di immagini di animali.
- Fare in modo che gli algoritmi di machine learning elaborino quelle immagini.
- Provare questi algoritmi su una serie di immagini sconosciute.
- Capire perché alcuni risultati non sono accurati.
- Migliorare il set di dati etichettando nuove immagini per migliorare l'accuratezza dei risultati.

Questo processo è chiamato apprendimento supervisionato. Nell'apprendimento supervisionato, l'accuratezza dei risultati migliora solo quando si dispone di un set di dati ampio e sufficientemente vario. Ad esempio, l'algoritmo potrebbe identificare accuratamente i gatti neri ma non i gatti bianchi perché il set di dati di addestramento conteneva più immagini di gatti neri. In tal caso, si dovrebbero etichettare più immagini di gatti bianchi e addestrare nuovamente i modelli di machine learning [19]. Esistono vari metodi per creare dei forti modelli di deep learning:

- **Transfer learning:** [20] Viene utilizzato per riaddestrare un modello di rete neurale, allo scopo di risolvere un problema simile a quello per cui è stato progettato, mediante l'utilizzo di uno o più livelli dello stesso. Il Transfer learning può essere molto utile nel caso in cui lo sviluppo del modello di rete neurale abbia portato a etichettare molti più dati rispetto al problema di iniziale interesse, e vi è una somiglianza nella struttura del nuovo problema da risolvere. L'obiettivo, dunque, è quello di sfruttare i dati della prima impostazione del modello neurale per estrarre informazioni che possono essere utili durante la fase di apprendimento del modello o per effettuare nuove previsioni in una seconda impostazione dello stesso.
- **Training from scratch:** [21] Consente agli sviluppatori di configurare da zero le architetture di rete raccogliendo grandi quantità di dati etichettati. Questo è il metodo meno utilizzato poiché l'addestramento richiede molto tempo a causa della grande quantità di dati. Rispetto ai metodi precedenti, richiede più dati e più tempo di elaborazione per ottenere risultati comparabili.
- **Dropout:** [22] Questa tecnica tenta di risolvere il problema dell'overfitting nelle reti con una grande quantità di dati, eliminando i neuroni dalla rete neurale durante l'addestramento, in altre parole, diversi neuroni vengono rimossi temporaneamente dalla rete. Durante l'addestramento, il dropout modifica l'idea di apprendere tutti i pesi nella rete, per apprendere solo una frazione dei pesi nella rete. Dopo ogni iterazione, vengono attivati diversi gruppi di neuroni, in modo da evitare che alcuni neuroni dominino il processo. Questo, quindi, ci aiuta a ridurre la minaccia dell'overfitting e consente la nascita di architetture di rete più profonde e più grandi che possono fare buone previsioni su dati.
- **Few-shot learning:** [23] Permette allo sviluppatore di creare un dataset con pochi esempi, in modo da eliminare il problema della grandezza del dataset. L'obiettivo principale nei framework Few-Shot tradizionali è quello di apprendere una funzione di somiglianza, che sia in grado di mappare le somiglianze tra le classi nei set di supporto e di dati. Le funzioni di somiglianza generano un valore di probabilità per la somiglianza.

Il deep learning ha diversi casi d'uso nel settore ambientale, automobilistico, aerospaziale, manifatturiero, elettronico, della ricerca medica e in altri campi [19]. Ecco

alcuni esempi di deep learning: [19]

- I sistemi per l'analisi e l'interpretazione fenomenologica ambientale.
- Le auto a guida autonoma utilizzano modelli di deep learning per rilevare automaticamente segnali stradali e pedoni.
- I sistemi di difesa utilizzano il deep learning per segnalare automaticamente le aree di interesse nelle immagini satellitari.
- L'analisi delle immagini mediche utilizza il deep learning per rilevare automaticamente le cellule tumorali per la diagnosi medica.
- Le fabbriche utilizzano applicazioni di deep learning per rilevare automaticamente quando persone o oggetti si trovano a una distanza non sicura dalle macchine.

Si possono raggruppare i diversi casi d'uso del deep learning in quattro grandi categorie: visione artificiale, riconoscimento vocale, elaborazione del linguaggio naturale (NLP) e motori di raccomandazione. [19]

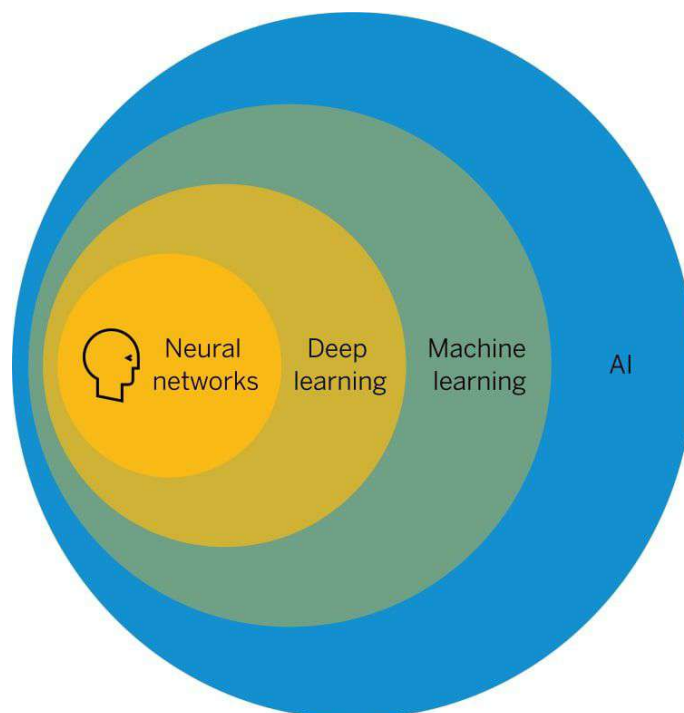


Figura 2.2: Branche dell'AI. [3]

2.2.1 Reti Neurali

I modelli di Deep Learning sono formati da delle reti neurali artificiali, che non sono altro che dei modelli matematici che rappresentano lo stesso funzionamento di una rete neurale che si trova all'interno di un cervello umano [24]. Il termine rete neurale viene utilizzato come riferimento a una rete o a un circuito formato da neuroni, ovvero unità funzionali del sistema nervoso, un neurone è una cellula altamente specializzata per ricevere, elaborare e trasmettere le informazioni ad altri neuroni o altre cellule, attraverso segnali elettrici e chimici. Il neurone è costituito da quattro componenti: il corpo cellulare (pirenoforo o soma), i dendriti, il cono di emergenza e l'assone. [24, 25]

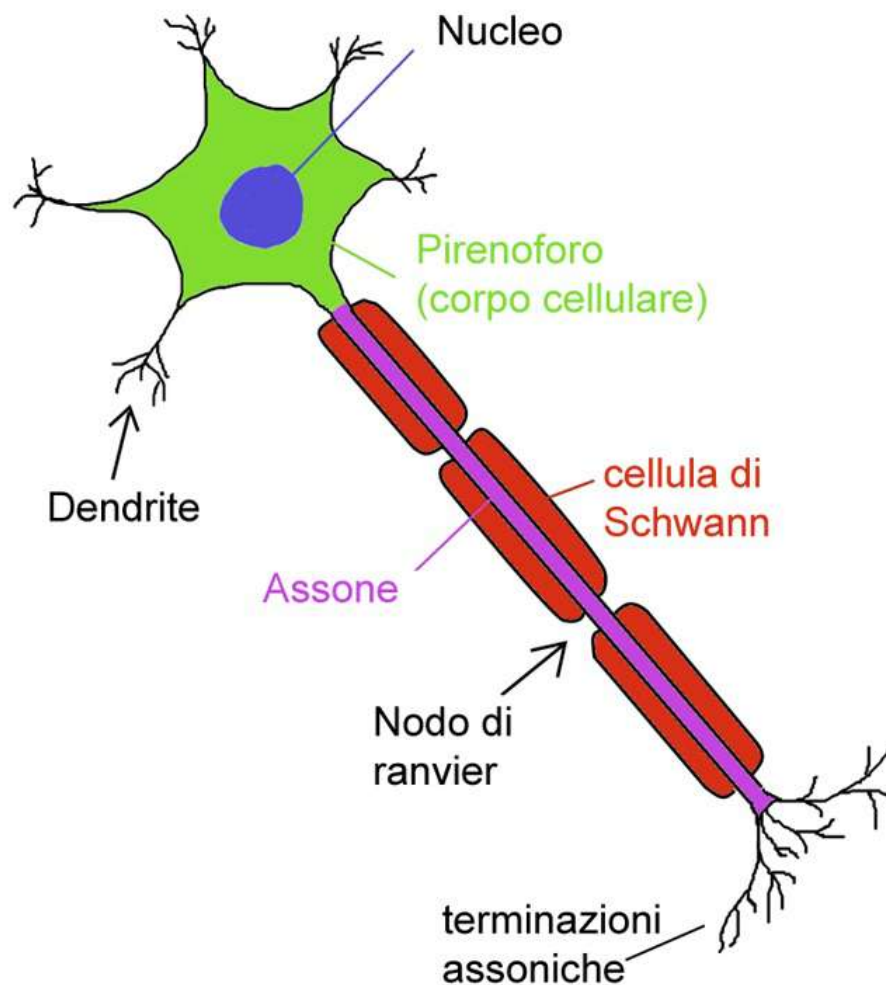


Figura 2.3: Neurone. [4]

In una rete neurale artificiale rappresentiamo un neurone nel seguente modo:

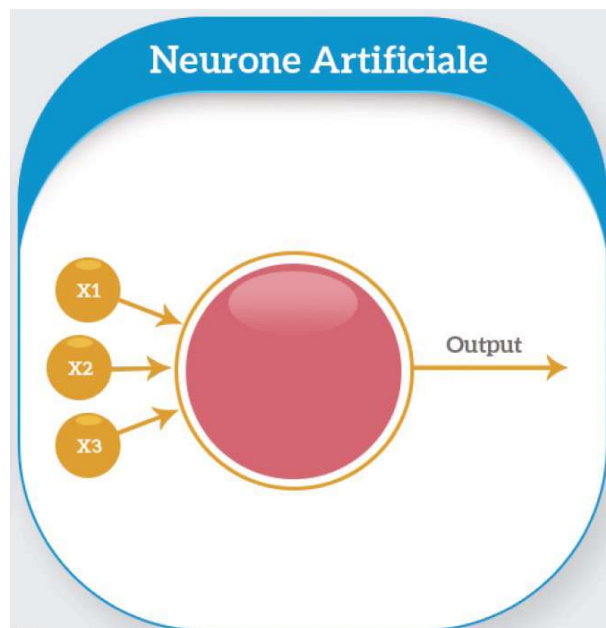


Figura 2.4: Neurone all'interno di una rete neurale artificiale. [5]

All'interno della rete neurale, ogni neurone ha dei collegamenti di input chiamati dendriti, associati a dei collegamenti di output (rappresentati da un singolo filamento chiamato assone). L'assone si divide, a sua volta, in diverse nervature che prendono il nome di terminali, che infine si collegheranno con i dendriti di un altro neurone [24]. Quindi otteniamo una rete neurale artificiale di questo tipo:

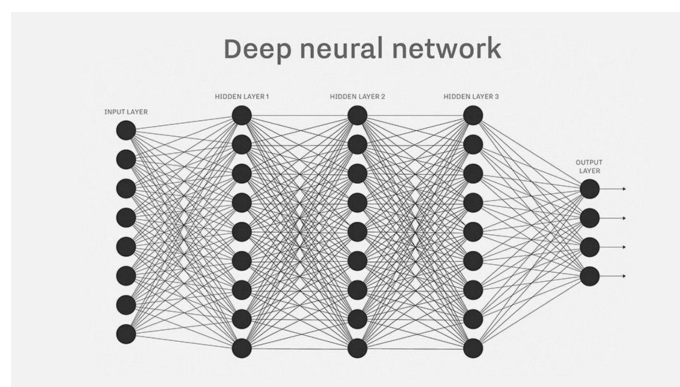


Figura 2.5: Rete neurale artificiale. [6]

I componenti di una rete neurale profonda sono i seguenti: [26, 19]

- Input layer

Una rete neurale artificiale ha diversi nodi che vi immettono dati. Questi nodi costituiscono il livello di input del sistema.

- Hidden layer

Il layer di input elabora e trasmette i dati ai livelli successivi nella rete neurale. Questi livelli nascosti (Hidden layer) elaborano le informazioni a diversi livelli, adattando il loro comportamento man mano che ricevono nuove informazioni. Le reti di deep learning hanno centinaia di livelli nascosti che possono utilizzare per analizzare un problema da diverse angolazioni.

- Output layer

Il livello di output è costituito dai nodi che generano i dati. I modelli di deep learning che generano risposte "sì" o "no" hanno solo due nodi nel livello di output. D'altra parte, quelli che producono una gamma più ampia di risposte hanno più nodi.

A differenza della programmazione "convenzionale", una rete neurale impara dai dati osservativi in una logica di "autoapprendimento", individuando la propria soluzione al problema. Prendendo in considerazione questa rete neurale:

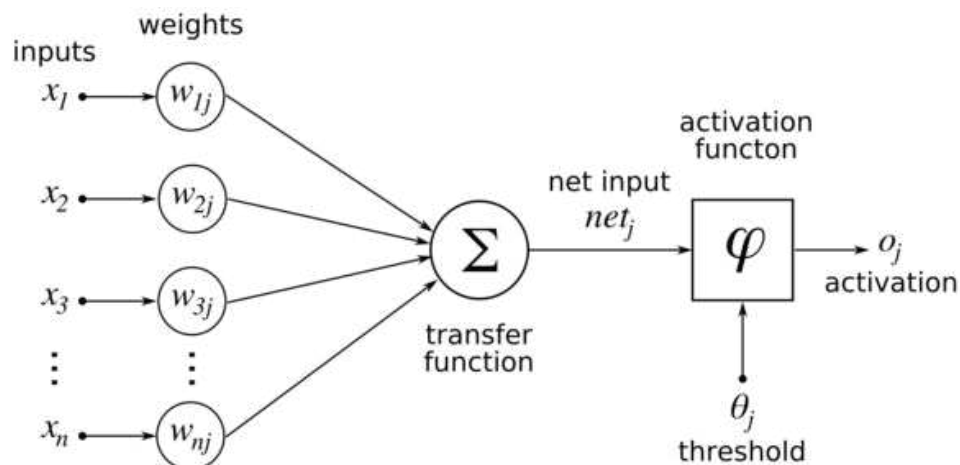


Figura 2.6: Funzione di attivazione. [7]

Possiamo notare come ad ogni neurone, sia associata una funzione che ne determina il funzionamento [26] (prende il nome di funzione di trasferimento [27]). Ogni

input da in output un determinato valore, quindi, il neurone prende in input un Z , dove Z non è altro che la sommatoria pesata per ogni input x più un certo bias. Questo valore Z verrà dato in input ad una funzione di attivazione del neurone successivo. La funzione di attivazione non è altro che una funzione che determina l'attivazione o meno di un neurone, quindi, se in quell'iterata(epoca) l'input è rilevante o meno. Data una funzione di attivazione $a(Z)$, essa darà un output che chiameremo \hat{y} , quindi:

$$\hat{y} = a(Z)$$

Il valore di output sarà input di un altro neurone e così via. Esistono diversi tipi di funzione di attivazione e si cerca di avere una non linearità di essa e la migliore dipende dal contesto di utilizzo. Un esempio di funzione di attivazione è la SIGMOID. [26, 28]

Una rete neurale potrebbe restituire un output differente da quello atteso, per cui essendo basata su una logica di autoapprendimento, cercherà ad ogni epoca di migliorare il suo output in modo da avvicinarsi a quello atteso. Per fare ciò, ci si basa su una funzione di loss detta anche funzione di perdita. La funzione di loss si occupa di determinare l'errore che porta ad un risultato sbagliato della predizione, il tutto solo se si conosce il valore reale di ciò che si deve predire. Essa definisce di quanto l'output è errato in quella specifica iterazione. Una rete neurale in generale non si basa su un'unica iterazione, poiché nell'iterazione successiva l'errore potrebbe essere maggiore di quello precedente. Chi si occupa di tener conto dell'errore di ogni epoca è la funzione di costo. Esistono diversi tipi di funzione di costo e di loss. In base all'errore calcolato, attraverso le funzioni di costo e di loss, una rete neurale tenta di cambiare i pesi w e il Bias b di ogni neurone per ogni epoca, poiché si pone l'obiettivo di minimizzare l'errore nelle epoche successive. Il modo in cui la rete neurale tenta di cambiare i pesi w in funzione del costo viene calcolato attraverso delle derivate parziali, che in caso di miglioramento, cambierà i pesi w con i nuovi calcolati. La rete neurale si fermerà quando i pesi w , daranno un costo prossimo allo zero. Alcune tecniche di miglioramento delle reti neurali considerando le epoche sono la BackwardPropagation e la ForwardPropagation. [26, 28]

2.2.2 Vantaggi e difficoltà dell'utilizzo del Deep Learning

Una rete di deep learning offre diversi vantaggi rispetto al machine learning tradizionale, ma poiché è una tecnologia relativamente nuova, vi sono alcune difficoltà che derivano dalla sua implementazione pratica. [19]

Vantaggi: [19].

- **Elaborazione efficiente dei dati non strutturati:** I metodi di machine learning trovano i dati non strutturati, come i documenti di testo, difficili da elaborare perché il set di dati di addestramento può avere infinite variazioni. D'altra parte, i modelli di deep learning possono comprendere dati non strutturati e fare osservazioni generali senza l'estrazione manuale delle funzionalità. Ad esempio, una rete neurale può riconoscere che queste due diverse frasi di input hanno lo stesso significato:

Puoi dirmi come effettuare il pagamento?

Come trasferisco il denaro?

- **Relazioni nascoste e individuazione di modelli:** Un'applicazione di deep learning può analizzare grandi quantità di dati in modo più approfondito e rivelare nuove informazioni per le quali potrebbe non essere stata addestrata. Ad esempio, considerando un modello di deep learning addestrato per analizzare gli acquisti dei consumatori. Il modello ha dati solo per gli articoli che sono già stati acquistati. Tuttavia, la rete neurale artificiale può suggerire nuovi articoli che non hai acquistato confrontando i tuoi modelli di acquisto con quelli di altri clienti simili.

- **Elaborazione di dati instabili:** I set di dati instabili presentano grandi variazioni. Un esempio sono gli importi del rimborso di un prestito in banca. Una rete neurale di deep learning può classificare e ordinare anche quei dati, ad esempio analizzando le transazioni finanziarie e segnalandone alcune per il rilevamento delle frodi.

Difficoltà: [19].

- Grandi quantità di dati di alta qualità: Gli algoritmi di deep learning forniscono risultati migliori quando vengono addestrati su grandi quantità di dati di alta qualità. I valori anomali o gli errori nel set di dati di input possono influire in modo significativo sul processo di deep learning. Ad esempio, nel nostro esempio di immagini di animali, il modello di deep learning potrebbe classificare un aereo come tartaruga se immagini non di animali venissero accidentalmente introdotte nel set di dati. Per evitare tali imprecisioni, prima di poter addestrare modelli di deep learning è necessario pulire ed elaborare grandi quantità di dati. La pre-elaborazione dei dati di input richiede grandi quantità di capacità di archiviazione di dati.

- Grande potenza di elaborazione: Gli algoritmi di deep learning richiedono una elaborazione intensiva e un'infrastruttura con capacità di calcolo sufficiente per funzionare correttamente altrimenti impiegheranno molto tempo per elaborare i risultati.

2.3 Modelli di linguaggio per la generazione del testo

Un modello di linguaggio è un modello statistico che modella la distribuzione delle sequenze di parole, più in generale di sequenze di simboli discreti (lettere, fonemi, parole), in un linguaggio naturale. Un modello linguistico può, ad esempio, prevedere la parola che segue una sequenza di parole. [29]

I modelli di linguaggio possono essere usati per task come ad esempio:

- Machine Translation: [30] è utile assegnare una probabilità a una frase per avere traduzioni più corrette.
- Spell Correction: [31] è possibile usare una probabilità assegnata a una frase per correggere eventuali errori di ortografia.
- Text Generation: [18] è utile calcolare la probabilità della prossima parola in una sequenza, per ottenere la probabilità di una sequenza di parole e così generare una frase di senso compiuto.

Alcuni modelli di linguaggio per la generazione del testo sono:

- Modelli di linguaggio basati su n-grams. [32]
- Modelli di linguaggio basati su Deep Learning. [33]

I modelli di linguaggio più efficienti sono quelli basati su Deep Learning, poiché più accurati, grazie all'utilizzo di reti neurali. I modelli di linguaggio basati su Deep Learning possono essere utilizzati per catturare efficacemente le dipendenze e le proprietà sequenziali di una sequenza di input. [33]

Più precisamente, essi sono adatti per la modellazione e la generazione del testo, poiché sono in grado di soddisfare i seguenti aspetti: [33]

- generazione automatica di caratteristiche.
- cattura delle dipendenze a lungo termine e delle proprietà sequenziali.
- apprendimento end-to-end.
- Generalizzabilità.

Mentre, nei modelli a n-grams, anche se possono estrarre automaticamente le caratteristiche dal testo, essi non possono catturare bene tutte le dipendenze a lungo termine a causa della complessità esponenziale del calcolo combinatorio. [32, 34]

2.3.1 Modello linguistico basato su n-grams

Un modello linguistico statistico semplice ed efficace è il modello di linguaggio basato su n-grams. Questo modello presuppone che ogni parola sia condizionatamente dipendente dalle $n - 1$ parole precedenti, attraverso la seguente formula: [34]

$$P(w_n | w_1, w_2, \dots, w_{n-1})$$

Questa formula rappresenta la probabilità condizionata. Il funzionamento di questo modello linguistico parte dalla suddivisione del testo in token¹. Se il token viene rappresentato con i singoli caratteri, si può immaginare come la complessità dell'algoritmo aumenti, in più, il carattere successivo potrebbe essere una ripetizione del

¹Un token è un'unità minima di analisi come ad esempio un carattere oppure una parola.

precedente. Se il token viene suddiviso in parole, allora la formula risulta essere molto efficiente sulla successiva parola da predire. Un vantaggio diretto degli n-grams è che sono facili da generalizzare. Gli n-grams non sono in grado di modellare le dipendenze a lungo termine tra i token, poiché si applica la formula su tutti i token del testo, come anche gli altri modelli statistici. Una possibile soluzione sarebbe troncare le informazioni posizionali a lungo termine, ovvero applicare la formula, ad esempio, solo agli ultimi 6-7 token inseriti nel testo (per ogni token). Inoltre, un'altra limitazione di un modello a n-grams è la scarsità della rappresentazione vettoriale della parola. Questo problema di scarsità può essere risolto utilizzando la rappresentazione distribuita dei modelli di linguaggio basati su Deep Learning. [34, 32]

2.3.2 Modelli linguistici basati su Deep Learning

I modelli linguistici basati su architetture neurali (neural networks) sono addestrate a predire parole mascherate in miliardi di frasi. Durante questo esercizio di predizione, le reti neurali regolano i propri parametri interni così da rappresentare nella propria struttura profonda, sia le relazioni tra le parole che il loro significato. Ed è proprio attraverso il linguaggio utilizzato che questi modelli linguistici imparano aspetti importanti della realtà [33]. Tali modelli possono usufruire di architetture di reti neurali di diverso tipo, come ad esempio: [33]

- FNN (Feedforward neural network): è una rete neurale artificiale dove le connessioni tra i nodi non formano cicli. Questo tipo di rete neurale fu la prima e la più semplice tra quelle messe a punto. In questa rete neurale le informazioni si muovono solo in una direzione, avanti, rispetto a nodi d'ingresso, attraverso nodi nascosti (se esistenti) fino ai nodi d'uscita. Nella rete non ci sono cicli. Le reti feed-forward non hanno memoria degli input avvenuti a tempi precedenti, per cui l'output è determinato solamente dall'attuale input. [35]
- RNN (Recurrent neural network): è una classe di rete neurale artificiale che include neuroni collegati tra loro in un ciclo. Tipicamente i valori di uscita di uno strato di un livello superiore sono utilizzati in ingresso di uno strato di livello inferiore. [36]

- LSTM (Long short-term memory): è una rete neurale artificiale utilizzata nei campi dell'intelligenza artificiale e del deep learning. A differenza delle reti neurali feedforward standard, LSTM ha connessioni di feedback. [37]
- Transformer: è un modello di apprendimento profondo che adotta il meccanismo dell'auto-attenzione, ponderando in modo differenziale il significato di ciascuna parte dei dati di input. [38]

Un modello di linguaggio basato su Deep Learning è la serie GPT di OpenAI. [29]

2.3.2.1 FNN

Una rete neurale feedforward è un tipo di rete neurale artificiale in cui le connessioni dei nodi non formano un ciclo. Spesso sono definite come una rete multistrato di neuroni. Le reti neurali feedforward sono così chiamate perché tutte le informazioni fluiscono solo in avanti. I dati entrano nei nodi in input, attraversano gli hidden layer ed infine escono dai nodi di output. La rete è priva di collegamenti che permettano alle informazioni che escono dal nodo di output di essere inviate nuovamente alla rete. Lo scopo delle reti neurali feedforward è quello di approssimare funzioni. Le prime architetture utilizzate per la generazione di testo furono le feedforward, queste architetture presentavano alcuni svantaggi come ad esempio:

La taglia dell'input e dell'output: esse sono definite a priori prima di utilizzare la rete neurale, per cui, va definita prima la struttura statica. Immaginiamo di risolvere un problema in cui gli input sono i pixel di un'immagine, sapendo che non tutte le immagini hanno lo stesso numero di input. Ipotizzando di definire staticamente un input pari a 1080, ovvero 1080 pixel e considerando un'immagine da 480 pixel, allora quali valori verranno assegnati ai restanti pixel? Oppure, immaginiamo di avere un testo, esso può avere un numero variabile di caratteri, per cui, non possiamo definire un numero di input statico nella rete neurale.

Infine, per ottenere un miglioramento di prestazioni dei modelli di linguaggio, basati su deep learning, si può fare riferimento su architetture come RNN, LSTN e Transformer. [35, 39]

2.3.2.2 RNN

Recurrent Neural Network (RNN) è un tipo di rete neurale in cui l'output dello step precedente viene inviato come input allo step corrente. Nelle reti neurali tradizionali, tutti gli input e gli output sono indipendenti l'uno dall'altro, ma in casi come quando è necessario prevedere la parola successiva di una frase, sono richieste le parole precedenti e quindi è necessario ricordarle. Così è stata sviluppata la rete RNN, che ha risolto questo problema con l'aiuto di uno hidden layer. La caratteristica principale e più importante di RNN è lo stato nascosto, che ricorda alcune informazioni su una sequenza. RNN ha una "memoria" che ricorda tutte le informazioni su quanto è stato calcolato. Utilizza gli stessi parametri per ogni input poiché esegue la stessa attività su tutti gli input o hidden layer per produrre l'output. Ciò riduce la complessità dei parametri, a differenza di altre reti neurali. L'idea principale delle RNN è quella di elaborare in modo efficiente i dati sequenziali, poiché è fondamentale per prevedere i risultati in modo più preciso, soprattutto in ambito di NLP. Attraverso le RNN si risolve il problema della size dell'input e dell'output, infatti, permette di definirla in maniera ottimale. In genere, una rete neurale tradizionale elabora l'input e passa al successivo senza considerare alcuna sequenza. I dati sequenziali, invece, vengono elaborati seguendo un ordine specifico, necessario per comprenderli in modo distinto. Inoltre, sapendo che le RNN sono ottime per essere utilizzate per problemi che riguardano lunghe sequenze di input, si utilizza il metodo del gradiente (ovvero la derivata parziale, che permette di calibrare i pesi e il bias per il miglioramento della rete neurale). La RNN assegna lo stesso peso e lo stesso bias a ciascuno degli strati della rete. Pertanto, tutte le variabili indipendenti vengono convertite in variabili dipendenti. I loop della RNN garantiscono la conservazione delle informazioni nella sua memoria. Questo meccanismo viene denominato backpropagation ed è stato una grande aggiunta alla procedura di addestramento. L'obiettivo dell'utilizzo della backpropagation è quello di ripercorrere la rete neurale in modo da identificare qualsiasi derivata parziale dell'errore rispetto ai pesi. Un problema delle RNN è che sono difficili da addestrare, a causa del problema del vanishing gradient. Il problema del vanishing gradient avviene quando la norma del gradiente è molto piccola e l'aggiornamento dei pesi è inefficace, rendendo impossibile per il modello

apprendere la correlazione tra eventi temporalmente lontani, quindi non contribuisce al miglioramento dell'apprendimento della rete neurale. Inoltre, non sono in grado di conservare le informazioni passate su diverse scale temporali. Una specializzazione di questa architettura sono le LSTM che tentano di risolvere questo problema. [36, 40]

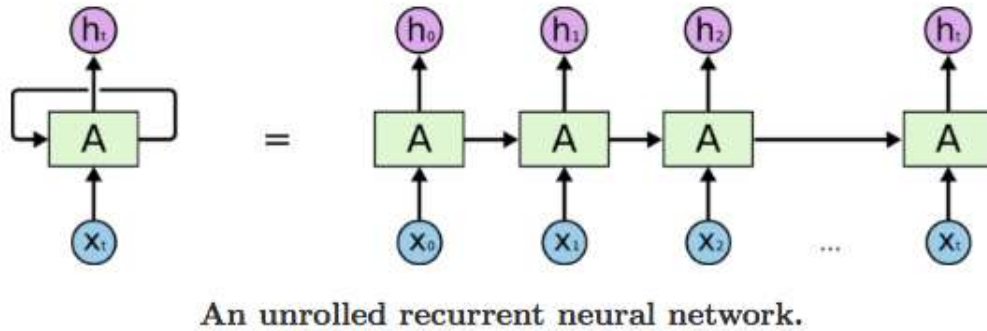


Figura 2.7: Recurrent Neural Network (RNN). [8]

2.3.2.3 LSTM

Una rete Long Short-Term Memory (LSTM) è una versione aggiornata della rete Recurrent Neural Network (RNN), per superare il problema del vanishing gradient. Una rete LSTM dispone di tre porte per controllare rispettivamente l'ingresso, l'uscita e la dimenticanza. Inoltre, c'è una cella di memoria che aiuta a trasportare le informazioni da una particolare istanza temporale all'istanza temporale successiva in modo efficiente. Quindi, può ricordare molte informazioni dagli stati precedenti rispetto alle reti RNN e superare il problema del vanishing gradient. Le reti LSTM sono comunemente usate nelle attività di NLP perché possono imparare il contesto richiesto per processare sequenze di dati. Le reti LSTM sono alimentate dai dati di input dall'istanza temporale corrente e dall'output del hidden layer dall'istanza temporale precedente.

LONG SHORT-TERM MEMORY NEURAL NETWORKS

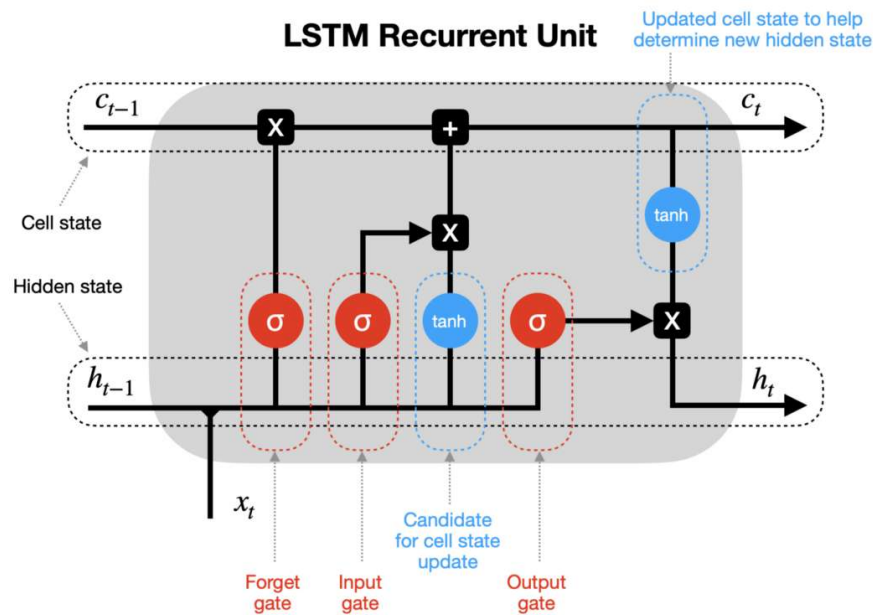


Figura 2.8: Long Short-Term Memory (LSTM). [9]

Possiamo notare che le reti LSTM sono molto simili alle reti RNN. Una differenza sostanziale con le reti RNN, è che ad ogni collegamento, vengono definiti dei gate (porte) che servono a definire dei calcoli precedenti, prima di far entrare un input all'interno di un neurone, per poi procedere con la funzione di attivazione. Questi calcoli, si occupano di determinare ciò che deve ricordare la rete neurale. Quindi si determina in quell'epoca, tutte quelle informazioni passate che la rete deve ricordare e quelle da dimenticare (informazioni meno rilevanti). Nell'NLG, le architetture LSTM, hanno una complessità computazionale maggiore, dato che una rete del genere risulta essere molto difficile da addestrare e parallelizzare, per cui subentrano le cosiddette architetture "Transformer". [37, 41]

2.3.2.4 Transformer

I transformer sono un particolare tipo di rete neurale che apprendono il contesto e quindi il significato tracciando le relazioni in dati sequenziali. I modelli Transformer applicano una serie di tecniche matematiche in evoluzione, chiamate attention o self-attention, per individuare i modi in cui elementi di dati in una serie, anche distanti tra loro, si influenzano e dipendono l'uno dall'altro. Notiamo nella Figura 2.9 come una

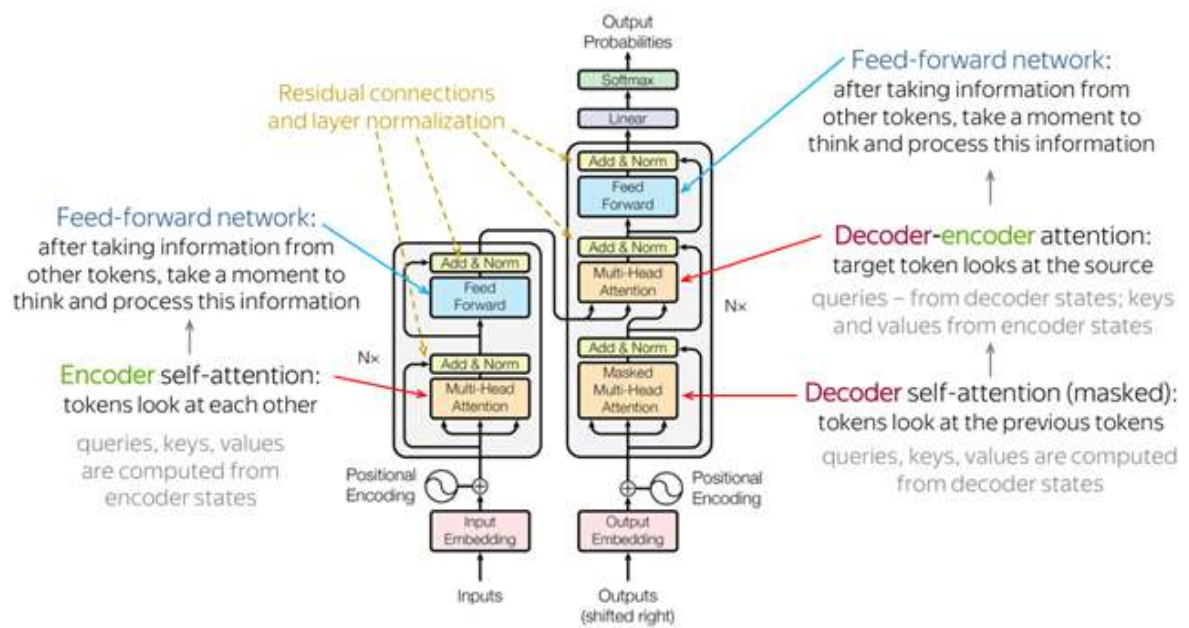


Figura 2.9: Rappresentazione di una rete Transformer. [10]

architettura di rete neurale transformer è composta da due componenti principali: encoder e decoder. L'encoder mappa la sequenza di simboli in input (x_1, \dots, x_n) in una sequenza di rappresentazioni continue $z = (z_1, \dots, z_n)$. Data la sequenza z , il decoder genera una sequenza di output (y_1, \dots, y_m) di simboli, generando un elemento alla volta. Ad ogni passo il modello è auto-regressivo ed utilizza i simboli generati precedentemente come un input aggiuntivo per produrre il prossimo. Come le RNN, i transformer sono utili per processare dati di input sequenziali, solo che a differenza delle reti RNN, processano i dati in una volta sola. Questo avviene attraverso la self-attention, che permette al trasformatore di non elaborare una parola alla volta. Ciò consente una maggiore parallelizzazione rispetto alle RNN e quindi riduce i tempi di addestramento. I transformer sono emersi grazie ai miglioramenti che hanno apportato in termini di efficienza e accuratezza nella gestione di task riguardanti il

Natural Language Processing. Queste reti non solo apprendono ripetendo le stesse azioni ma trovano anche pattern nei dati, imparando dal contesto con lo scopo di creare nuove informazioni. I transformer hanno cambiato considerevolmente il modo con cui lavoriamo con i dati testuali, tuttavia presentano alcune limitazioni. In particolare, l'attention può gestire solo stringhe di lunghezza fissata. Il testo viene quindi suddiviso in segmenti prima di essere inviato al sistema come input. È proprio questo accorciamento del testo che può portare alla frammentazione contestuale, con conseguente scissione di parti importanti del testo. Questo tipo di rete neurale ha portato alla realizzazione di modelli come: GPT, BERT... [38, 42]

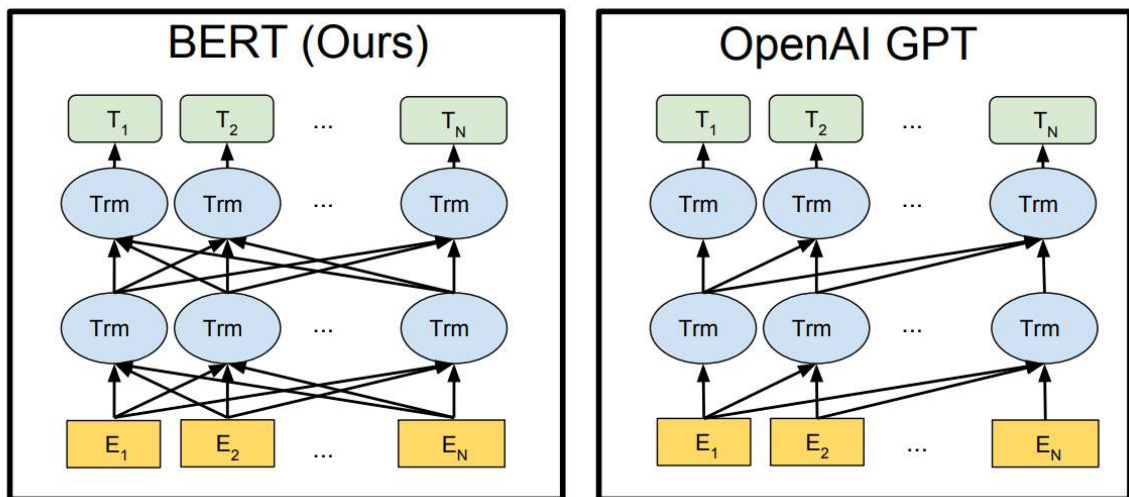


Figura 2.10: Modelli GPT, BERT. [11]

2.4 GPT-2

GPT-2 è l'acronimo di Generative Pre-trained Transformer 2, ed è un'intelligenza artificiale open source creata da OpenAI, che si occupa di generare testo a partire da input. Essa risulta essere molto utile anche per generare summary, tradurre del testo e rispondere a delle domande. GPT-2 è la seconda generazione della serie GPT. GPT è uno dei modelli più importanti e fondamentali per la comprensione del linguaggio, che ha contribuito a gettare le basi della modellazione linguistica. Questo modello è anche uno dei pionieri della nascita della NLP, per quanto riguarda l'elevato numero di parametri di addestramento, con 110 milioni di parametri (che ad oggi possono

sembrare pochi, ma all'epoca della sua uscita erano davvero tanti), ottenuto grazie all'utilizzo di una delle prime architetture Transformer. GPT si basa essenzialmente sul concetto di pre-addestramento e di transfer learning di un modello linguistico su un enorme corpus di dati e di successiva messa a punto. GPT-2 risulta essere più performante rispetto al suo predecessore, infatti, risulta essere 10 volte migliore di GPT, poiché offre 1,5 miliardi di parametri di addestramento ed inoltre è pre-addestrato su un dataset di 40GB. Esistono diverse varianti di GPT-2, ognuna suddivisa per lo spazio di memoria che occupa per memorizzare il modello sulla base dei suoi parametri. La variante più piccola di GPT-2 addestrato, occupa 500 MB di spazio di archiviazione per memorizzare tutti i suoi parametri. Mentre l'ultima variante di GPT-2 è 13 volte più grande, quindi potrebbe occupare più di 6,5 GB di spazio di archiviazione. [43, 12]



Figura 2.11: Varianti del modello di GPT-2. [12]

Per ottenere un miglioramento delle prestazioni, OpenAI ha sviluppato una nuova generazione di GPT, chiamata GPT-3. GPT-3 utilizza la stessa tipologia di architettura di GPT-2, ma la sostanziale differenza è che offre 175miliardi di parametri di addestramento contro i 1,5miliardi di GPT-2, ed è pre-addestrato su 45TB di testo rispetto ai 40GB del suo predecessore, inoltre, un altro suo vantaggio è la possibilità di poter effettuare il Few-shot learning. Infine, GPT-3 è closed source. [44]

2.4.1 Architettura e funzionamento

[12] GPT-2 si basa su un'architettura di rete neurale di tipo Transformer. Nell'architettura dei Transformer classica, essa è composta da encoder e decoder, infatti:

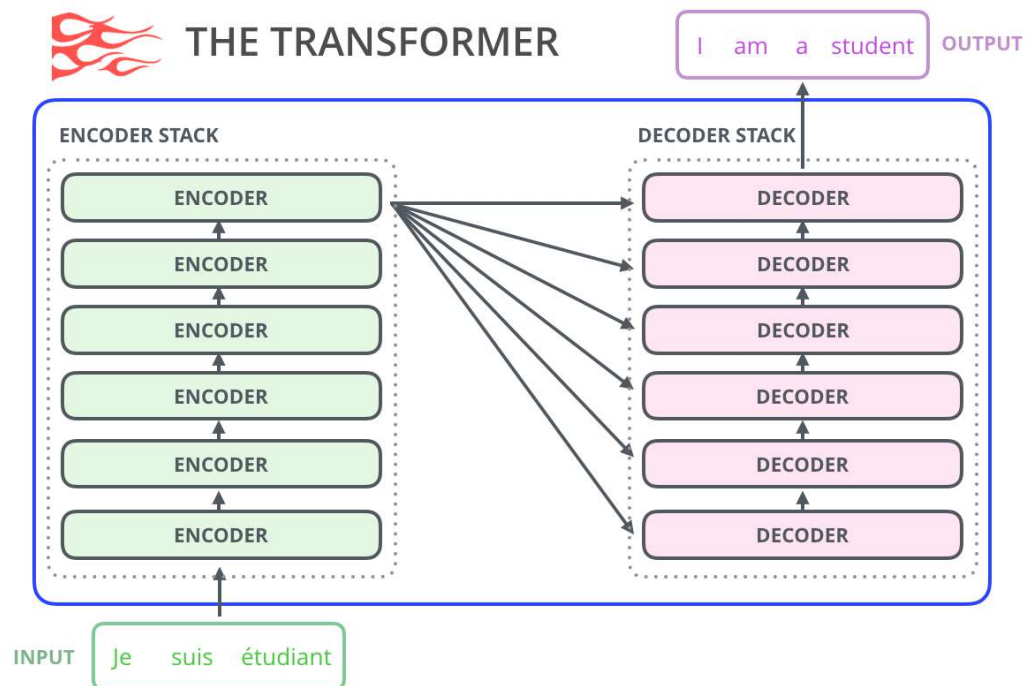


Figura 2.12: Architettura transformer. [12]

Gran parte del lavoro di ricerca successivo, ha visto l'architettura liberarsi di uno dei due blocchi, encoder decoder, ottenendo:

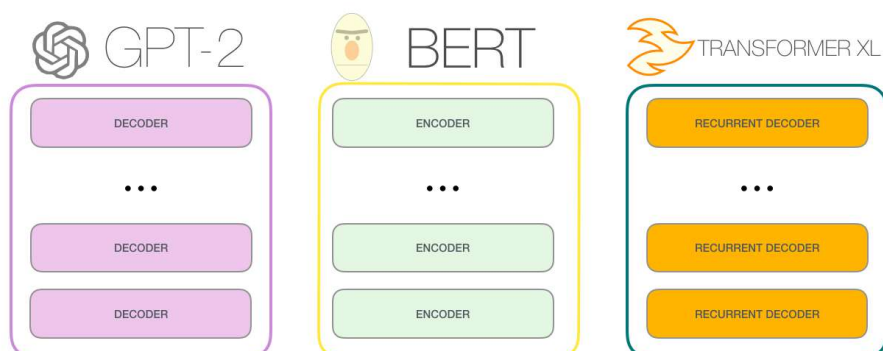


Figura 2.13: Modelli GPT, BERT, TRANSFORMER XL. [12]

Possiamo notare che una differenza architetturale tra GPT-2 e BERT sono i blocchi di encoder e decoder, infatti GPT-2 usa solo blocchi di decoder, mentre BERT solo blocchi di encoder. Inoltre, il blocco dei decoder e degli encoder di BERT e GPT-2 sono diversi dai blocchi classici, poiché, si è avuto un'evoluzione dei blocchi delle architetture Transformer, che risultano essere:

- Per gli encoder:

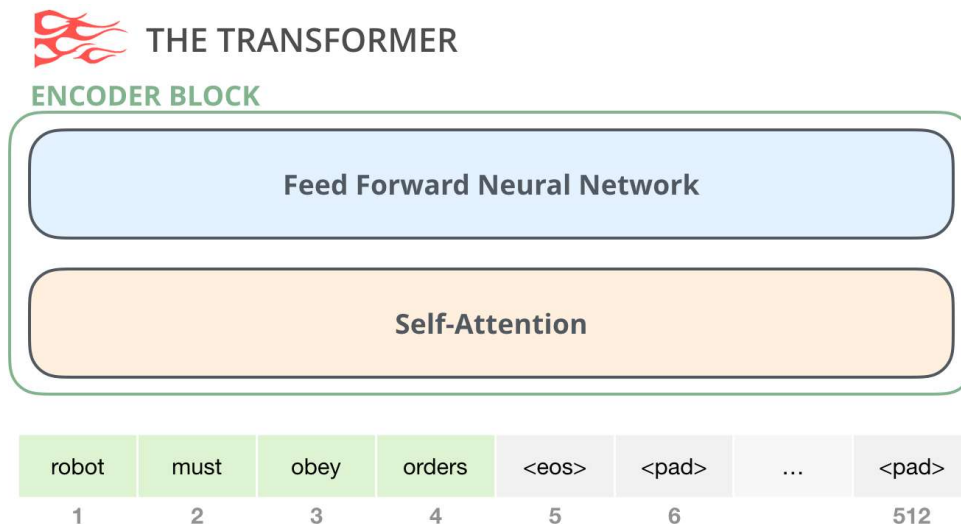


Figura 2.14: Blocco Encoder. [12]

- Per i decoder:

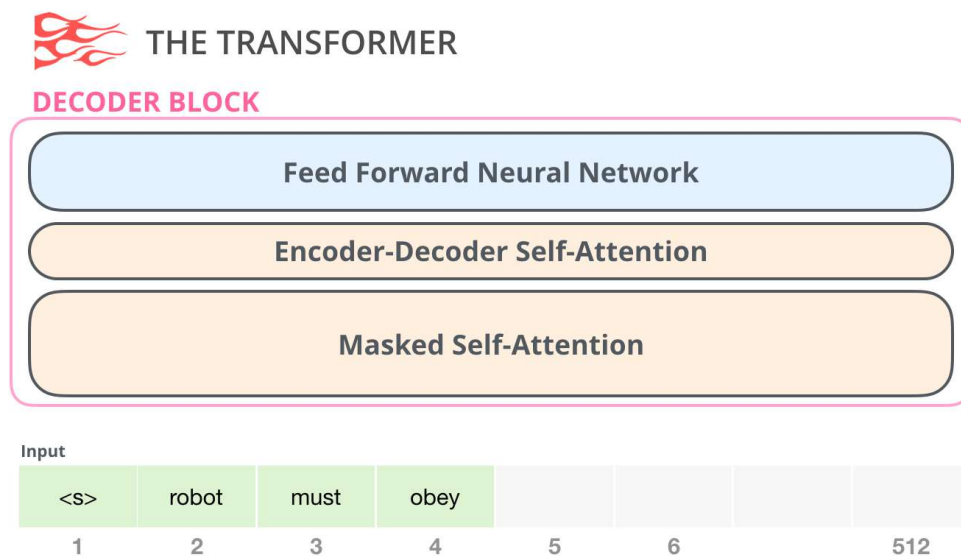


Figura 2.15: Blocco Decoder. [12]

Una differenza sostanziale è che GPT-2 costruisce la frase un token alla volta. Dopo che un token è stato generato, otteniamo una frase parziale, che sarà considerata come un nuovo input per il modello nella fase successiva. Il nuovo input sarà processato per generare un nuovo token, che verrà aggiunto come ultima parola, in modo da ripetere il processo, ottenendo una frase finale. Quest'idea prende il nome di auto-regressione. Ad esempio, GPT-2 genera la parola "PIANO" dopo un input, la parola "PIANO" sarà considerata come nuovo input. GPT-2 genererà un'altra parola come ad esempio "FORTE", concatenando "FORTE" e "PIANO", si ottiene "PIANOFORTE", che sarà un nuovo input del modello, in modo da generare il prossimo token e di concatenarlo alla frase di input, infine si ripete il processo.

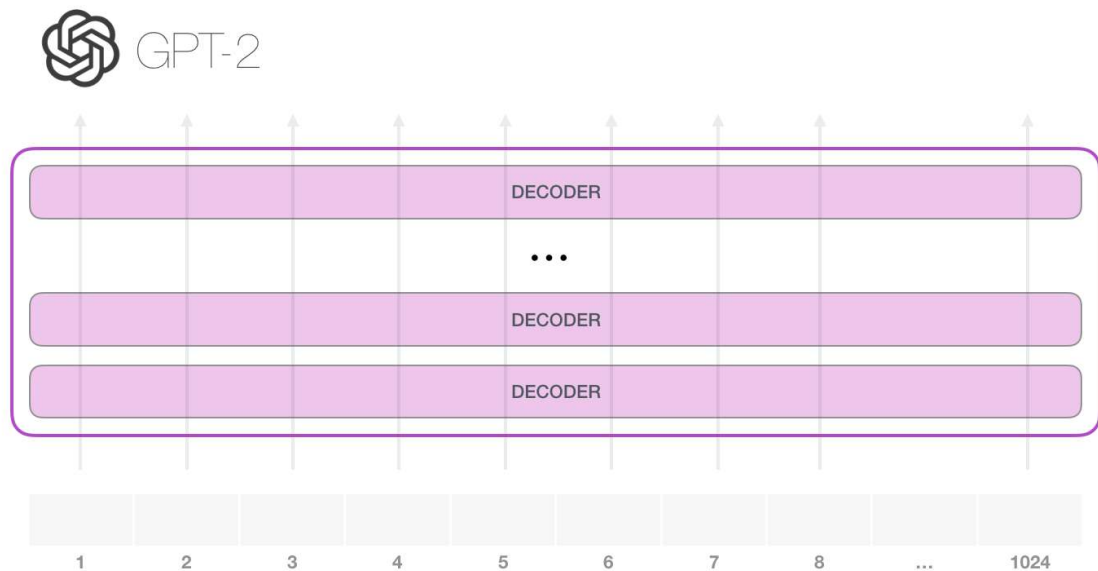


Figura 2.16: Generazione token(1). [12]

Come si può notare dalla Figura 2.16, il numero massimo di token che GPT-2 può utilizzare è 1024. Per la predizione di un token si utilizza la masked self-attention, ovvero per generare un token ci si basa su tutti i token precedenti ad esso. Più in dettaglio per la predizione di un nuovo token, tutti i token precedenti sono stati dati in input a tutti i decoder.

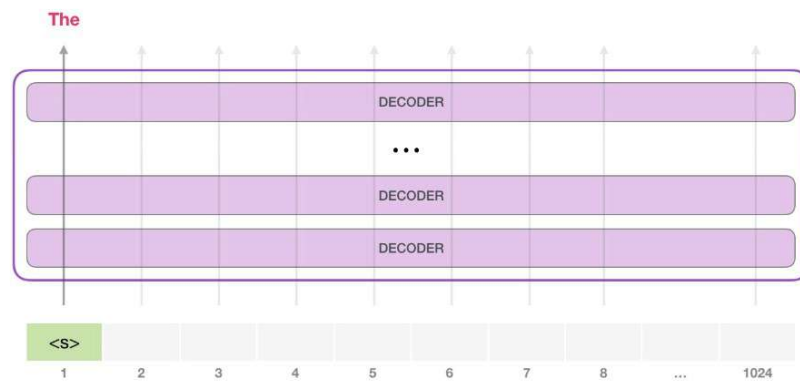


Figura 2.17: Generazione token(2). [12]

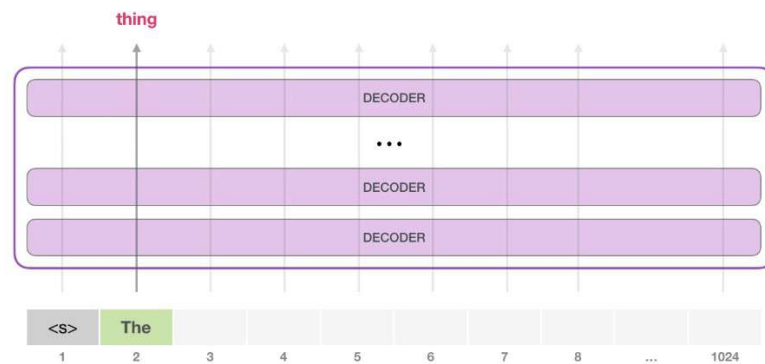


Figura 2.18: Generazione token(3). [12]

La masked self-attention del blocco dei Decoder si differisce dalla Self-Attention utilizzata dal blocco degli Encoder. Infatti, supponiamo di avere massimo 1024 token sia su GPT-2 che su BERT, e supponiamo di aver riempito solo i primi 4 token, ovvero quei token che mi rappresentano la frase parziale nella quinta iterata. GPT-2, attraverso l'utilizzo della masked self-attention, va a mascherare tutti i token, in modo da permettere la generazione della nuova parola nella quinta iterazione applicando la self-attention, ovvero il calcolo della previsione per la generazione della nuova parola, tenendo conto solo dei primi 4 token. BERT invece, andrà ad applicare la self-attention su tutti i token. Questo però potrebbe influenzare negativamente il calcolo della previsione della generazione della nuova parola.

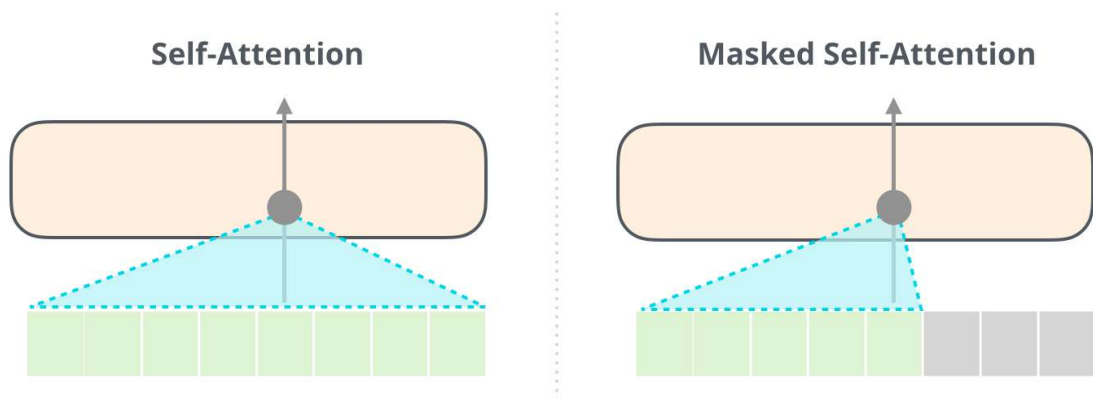


Figura 2.19: Differenze tra Self-Attention e Masked Self-Attention. [12]

Il blocco decoder, in generale, dispone di uno strato successivo alla masked self-attention, che prende il nome di encoder-decoder-self-attention. Questo strato si basa su una self-attention ottenuta sia dall'encoder che dal decoder. Esso viene utilizzato solo nell'architetture dei Transformer, che hanno sia blocchi di encoder che di decoder. Nel caso di GPT-2 non avendo Encoder, questo strato viene eliminato. Infine, per ogni blocco encoder | decoder si ha una rete neurale del tipo FFNN.

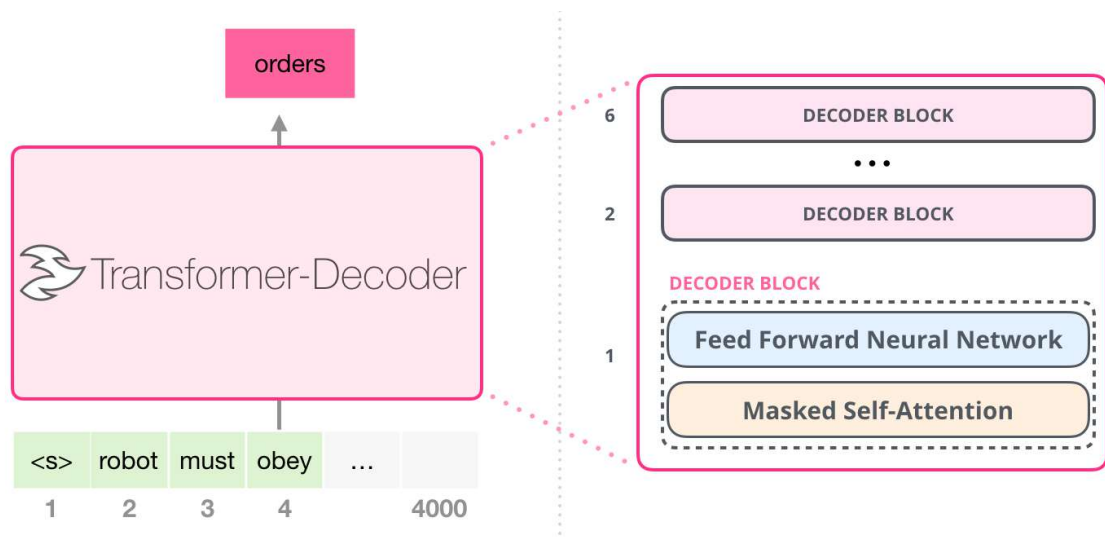


Figura 2.20: Architettura GPT-2. [12]

[13] GPT-2-Simple è una libreria custom che racchiude gli script esistenti per la messa a punto e la generazione del modello GPT-2 di OpenAI (in particolare le versioni "piccola" da 124M e "media" da 355M iperparametri). Inoltre, questo pacchetto consente una generazione più semplice del testo, la generazione in un file per una facile gestione, consentendo l'uso di prefissi per forzare il testo a iniziare con una determinata frase. Questo pacchetto incorpora e apporta modifiche minime di basso livello a:

- Gestione del modello dal repo ufficiale GPT-2 di OpenAI (licenza MIT)
- Finetuning del modello dal fork di Neil Shepperd di GPT-2 (licenza MIT)
- Gestione dell'output di generazione del testo da textgenrnn (Licenza MIT)

Per la messa a punto è fortemente consigliato l'uso di una GPU, anche se è possibile generare usando una CPU (il processo è più lento). Un'altra libreria di supporto per GPT-2-simple è TensorFlow. TensorFlow è una libreria software open source per l'apprendimento automatico, che fornisce moduli sperimentali e ottimizzati, utili nella realizzazione di algoritmi per diversi tipi di compiti percettivi e di comprensione del linguaggio. Questa libreria svolge diverse funzioni di supporto come ad esempio:

- Il salvataggio del modello in un determinato checkpoint.
- Il caricamento di un modello in un determinato checkpoint.
- possibilità di utilizzare la potenza computazionale della gpu per un aumento di prestazioni.

Lo sviluppo di gpt-2-simple è stato in gran parte superato da aitextgen, che ha capacità simili di generazione di testo AI con tempi di addestramento e utilizzo delle risorse più efficienti. Se non si desidera utilizzare TensorFlow, è consigliabile utilizzare aitextgen. I checkpoint addestrati con gpt-2-simple possono essere caricati anche con aitextgen.

2.4.1.1 Utilizzo di GPT-2-simple

[13] Per importare la libreria GPT-2-simple in python si utilizza il seguente comando:

```
1 import gpt_2_simple as gpt2
```

Per eseguire il download del modello 774M di GPT-2 si utilizza:

```
1 gpt2.download_gpt2("774M")
```

il modello viene salvato nella directory corrente con una specifica struttura.

Le funzioni di GPT-2-simple hanno bisogno di una sessione TensorFlow, che la si ottiene nel seguente modo:

```
1 sess = gpt2.start_tf_sess()
```

Il comando per eseguire il finetuning è il seguente:

```
1 gpt2.finetune(sess, 'dataset.txt', steps=1000)
2 # steps is max number of training steps
```

- sess: sessione tensorflow;
- dataset.txt: risulta essere il file di addestramento.
- steps: indica il numero di epoche di training da effettuare. Un'epoca di training si realizza quando il modello viene allenato su tutte le istanze del dataset una volta.

Inoltre, ci sono anche altri parametri come:

- Model_name: è il nome del modello utilizzato.
- Restore_from: può avere due valori, "fresh" e "latest" e indica, nel primo caso, se il modello debba ricominciare da zero il training oppure, nel secondo caso, se deve ripartire dall'ultimo checkpoint memorizzato.
- Run_name: indica il path dell'ultimo checkpoint salvato.
- Print_every: indica il numero di steps dopo il quale stampa i progressi fatti dal

modello riguardo al training. In particolare, stampa i valori di loss e loss avg che indicano, rispettivamente, i valori correnti e la media dei valori di loss.

- `Sample_every`: indica il numero di steps dopo il quale stampa una frase di prova generata senza nessun input.
- `Save_every`: indica il numero di steps dopo il quale il modello memorizza un checkpoint.

Una volta eseguito il finetuning del modello, esso potrà generare del testo. Per la generazione del testo si userà un comando del tipo:

```
1 single_text = gpt2.generate(sess, return_as_list=True) [0]
2 print(single_text)
```

Per ottenere un output più sistematico, si possono utilizzare diversi parametri come:

- `prefix`: indica il prefisso con cui dovrebbe iniziare la parola.
- `length`: indica la lunghezza dei token. GPT-2 ne mette a disposizione al più 1024.
- `temperature`: indica il grado di “fantasia” con cui il modello esegue gli accoppiamenti di parole per generare dei periodi originali. Questo valore è un float, ovvero un decimale, che può andare da 0.1 a 1, e i suoi valori consigliati sono: 0.5 per 60 parole, tra 0.6 e 0.8 per minimo 100 parole, e in generale valori maggiori di 0.6 per più di 100 parole con un training set molto grande.
- `nsamples`: indica il numero di esempi di frasi generate con lo stesso input.

Generazione di report ambientali tramite l'uso di intelligenza artificiale

In questo capitolo ci si addenterà nello sviluppo dell'intero sistema di intelligenza artificiale per la generazione di piccoli report ambientali, che descrivono in maniera accurata lo stato della qualità dell'aria ed eventi storici di una determinata provincia o regione. Il primo passo da eseguire sarà quello di costruire un dataset per il modello, composto da dati prettamente artificiali, che racchiudono una maggior parte di possibili combinazioni degli inquinanti scelti. Subito dopo, si prosegue con l'addestramento attraverso un modello basato su Transformers, in particolare si vedrà l'utilizzo di GPT-2, che si è mostrato nella sezione 2.4. La principale task del modello sarà di prendere in input una stringa e dare in output un'altra stringa, che sia conforme con le informazioni date in input, quindi, essere molto accurata con i dati ambientali. Creare un sistema di intelligenza artificiale a singolo layer, che generi frasi conformi con i dati ambientali in costante aggiornamento, non basta. L'intero sistema dovrà essere composto da due layer. Il primo layer servirà per la costruzione di un input sistematico, così che esso sia sempre aggiornato con i dati nazionali attraverso le opportune analisi. Nel secondo layer invece sarà implementato il nostro modello.

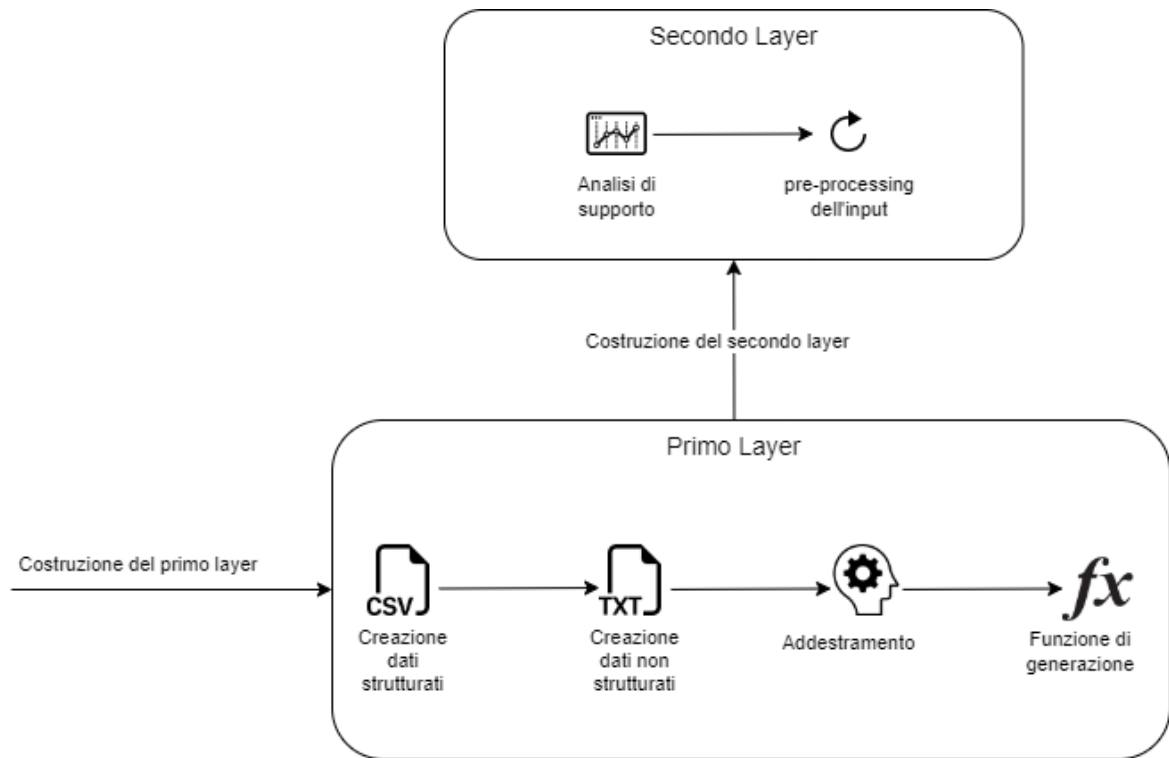


Figura 3.1: Pipeline eseguita.

La seguente immagine, mostra l'intera pipeline eseguita per la realizzazione dell'intero sistema. Si parte con la costruzione del primo layer, dedicato interamente ai dati e al modello. Inizialmente si realizza un dataset con dati strutturati in formato csv di cui se ne discute nella sezione 3.2, per poi convertirlo in un dataset con dati non strutturati in formato txt nella sezione 3.3. Una volta ottenuti tutti i dati di cui il modello necessita, si esegue l'addestramento nella sezione 3.4. Per concludere il primo layer, si è definita una funzione per la generazione dei report, che verrà richiamata da uno script main, tale funzione si mostra nella sezione 3.5. Di conseguenza, ci si concentra sulla costruzione del secondo layer (sezione 3.6), per poi creare uno script main per la messa in produzione (sezione 3.7).

3.1 Specifiche tecniche e librerie utilizzate

La libreria utilizzata per la costruzione del modello di GPT2 è GPT-2-SIMPLE¹. I vantaggi di questa libreria risiedono nel fatto che permette di effettuare il fine-tuning sul pre-trained nella maniera più semplice possibile. Essa può essere usata in maniera del tutto black box, infatti, ad esempio durante il fine-tuning, l'utilizzatore non dovrà nemmeno richiamare ulteriori metodi per l'uso del tokenizer. La libreria inoltre fornisce varie versioni di gpt2, che risultano essere: 124M, 355M, 774M e 1558M.

Tabella 3.1: Differenze delle versioni fornite da GPT-2-SIMPLE [13].

Versione	Peso	Fine-Tuning
124M	circa 500MB	SI
335M	circa 1.5GB	SI
774M	circa 3-4GB	SI
1558M	circa 6-7GB	NO

Nota: Per "Fine-Tuning" si indica se è possibile effettuare l'addestramento su una specifica versione, data la libreria in questione.

Come si può vedere dalla tabella, sulla versione 1558M non è possibile effettuare il fine-tuning, ed ecco perché la versione utilizzata nel nostro sistema risulta essere la 774M. Per poter utilizzare la versione 774M, si è utilizzati una macchina dotata di una CPU per server e di 64GB di RAM. La libreria GPT-2-SIMPLE può essere utilizzata anche attraverso l'aiuto di GPU con supporto driver CUDA, il che migliora le prestazioni in termini di velocità durante l'addestramento e la generazione. Per poter usufruire di una GPU, c'è ovviamente bisogno di un minimo di GB supportati.

¹<https://github.com/minimaxir/gpt-2-simple>

Il codice per usufruire di GPT-2-SIMPLE e della versione 774M risulta essere:

```
1 pip3 install gpt-2-simple
```

Tale istruzione è stata usata sulla riga di comando.

```
1 import gpt_2_simple as gpt2
2
3 gpt2.download_gpt2(model_name="774M")
```

Tale istruzione è stata usata per eseguire il download della versione 774M.

Altre librerie utilizzate nel sistema sono:

- TensorFlow è una libreria indispensabile che fa da supporto a GPT-2-SIMPLE, può essere utilizzata per decidere se integrare o meno l'uso della gpu e permette di creare sessioni.

```
1 pip3 install tensorflow.
```

- Pandas e Numpy sono librerie utilizzate per manipolare i dati strutturati in formato csv, utile anche per il parsing da dati strutturati a non strutturati sintetici.

```
1 pip3 install pandas
2 pip3 install numpy
```

3.2 Creazione dataset con dati strutturati

Per la creazione del dataset con dati strutturati, si è partiti da un primo set di dati in formato csv, contenente tutte le province associate alle rispettive regioni d'Italia, monitorate e fornite dall'azienda Sense Square. Subito dopo si è proceduti con l'inizializzazione del dataset con dati strutturati finale:

```
1 import pandas as pd
2 import numpy as np
3 import os
4 #inizializzo le colonne
5 preparazione = {
6     'PROVINCIA': [np.nan],
7     'REGIONE': [np.nan],
8     'PM10': [np.nan],
9     'MINMAX': [np.nan],
10    'SEASON': [np.nan],
11    'FREQ': [np.nan]
12 }
13
14
15 column_name=[ 'PROVINCIA', 'REGIONE', 'PM10', 'MINMAX', 'SEASON', 'FREQ' ]
16
17 df = pd.DataFrame(preparazione)
18 df.to_csv("dataset.csv", index=False, columns=column_name, encoding='utf-8')
```

Tale codice è stato utilizzato per inizializzare il dataset.

Dopo l’inizializzazione si è passati alla creazione del dataset finale con dati strutturati in formato csv, attraverso il seguente codice:

```
1 import pandas as pd
2 import numpy as np
3 import os
4
5 #mi creo tutte le possibili combinazioni in formato csv
6
7 provreg = pd.read_csv("provreg.csv")
8 limit = 15
9 column_name=['PROVINCIA','REGIONE','PM10','MINMAX','SEASON','FREQ']
10 season=['spring','summer','autumn','winter']
11
12 for index, row in provreg.iterrows():
13     for cuno in range(30):
14         for cdue in range(10):
15             for minmax in range(5):
16                 for sson in range(4):
17                     for freq in range(3):
18                         app = float(cuno + (cdue/10))
19                         preparazione = {
20                             'PROVINCIA' : [row['PROVINCIA']],
21                             'REGIONE': [row['REGIONE']],
22                             'PM10': [round(limit-app, 1)],
23                             'MINMAX': [minmax],
24                             'SEASON': [season[sson]],
25                             'FREQ': [freq]
26                         }
27                         df = pd.DataFrame(preparazione)
28                         df.to_csv('dataset.csv', mode='a',
29                             index=False, header=False)
```

In tale dataset è stato scelto come inquinante il pm10.

```
1 PROVINCIA,REGIONE,PM10,MINMAX,SEASON,FREQ
2 Salerno,Campania,15.0,0,spring,0
3 Salerno,Campania,15.0,0,spring,1
4 Salerno,Campania,15.0,0,spring,2
5 Salerno,Campania,15.0,0,summer,0
6 Salerno,Campania,15.0,0,summer,1
7 Salerno,Campania,15.0,0,summer,2
8 Salerno,Campania,15.0,0,autumn,0
9 Salerno,Campania,15.0,0,autumn,1
10 Salerno,Campania,15.0,0,autumn,2
11 Salerno,Campania,15.0,0,winter,0
12 Salerno,Campania,15.0,0,winter,1
13 Salerno,Campania,15.0,0,winter,2
14 Salerno,Campania,15.0,1,spring,0
15 Salerno,Campania,15.0,1,spring,1
16 Salerno,Campania,15.0,1,spring,2
17 Salerno,Campania,15.0,1,summer,0
18 Salerno,Campania,15.0,1,summer,1
19 Salerno,Campania,15.0,1,summer,2
20 Salerno,Campania,15.0,1,autumn,0
21 Salerno,Campania,15.0,1,autumn,1
22 Salerno,Campania,15.0,1,autumn,2
23 Salerno,Campania,15.0,1,winter,0
24 Salerno,Campania,15.0,1,winter,1
25 Salerno,Campania,15.0,1,winter,2
26 Salerno,Campania,15.0,2,spring,0
27 Salerno,Campania,15.0,2,spring,1
28 Salerno,Campania,15.0,2,spring,2
29 Salerno,Campania,15.0,2,summer,0
30 Salerno,Campania,15.0,2,summer,1
31 Salerno,Campania,15.0,2,summer,2
32 Salerno,Campania,15.0,2,autumn,0
```

Figura 3.2: Visualizzazione parziale del dataset finale con dati strutturati.

Come possiamo vedere dalla figura 3.2, abbiamo quasi tutte le possibili combinazioni di pm10 che potrebbe avere una specifica provincia di una specifica regione. Inoltre abbiamo anche tutte le possibili combinazioni di altri campi come:

- **MINMAX:** è un campo che può assumere valori da 0 a 4. Il valore 0 indica che il pm10 registrato risulta essere un nuovo valore di minimo per l'intera regione. Il valore 1 indica che il pm10 registrato è uguale o molto vicino al valore minimo dell'intera regione. Il valore 2 indica che si tratta di un risultato stabile di pm10 registrato. Il valore 3 indica che il pm10 registrato è uguale o molto vicino al valore massimo dell'intera regione. Il valore 4 indica che il pm10 registrato risulta essere un nuovo valore di massimo per l'intera regione.
- **SEASON:** è un campo che può assumere come valore il nome di una stagione. Sono state inserite tutte le combinazioni delle quattro stagioni per ogni qual volta si verifica un dato pm10 in una provincia.
- **FREQ:** è un campo che può assumere valori da 0 a 2. Il valore 0 indica che c'è poca frequenza di pm10 registrato in una data stagione (0-29%). Il valore 1 indica che il pm10 registrato in una data stagione è abbastanza frequente (30-59%). Il valore 2 indica che il pm10 registrato in una data stagione è molto frequente (60-100%).

Il dataset finale con dati strutturati, risulta avere un peso complessivo di 66.009KB. Esso dovrà essere processato per creare un nuovo dataset con dati non strutturati, necessario per l'addestramento del modello.

3.3 Creazione dataset con dati non strutturati

Dopo la costruzione del dataset con dati strutturati, bisogna convertire i dati da strutturati a non strutturati (frasi), creando l'effettivo dataset finale per l'addestramento del modello. Questa conversione porta alla creazione di frasi sintetiche e molto uguali tra loro, che siano strettamente legate ai dati ambientali.

```

1 import pandas as pd
2 import numpy as np
3 import os
4
5
6
7 def limitvalue(pm10):
8     str=''
9     if(not np.isnan(pm10)):
10         str = ' a PM10 equal to {}'.format(pm10)
11         if(pm10 >= 0):
12             str = str + " and it is within the limit value"
13         elif(pm10 < 0):
14             str = str + " and it exceeds the limit value"
15         str=str+" "
16     return str
17
18 def minmax(flag):
19     str=''
20     if(not np.isnan(flag)):
21         if(flag == 0):
22             str = str + " This PM10 value also turns out
23             to be a new minimum for the entire region"
24         elif(flag == 1):
25             str = str + " This PM10 value turns out
26             to be closer to the lowest value recorded
27             in the entire region"
28         elif(flag == 2):
29             str = str + " This PM10 value appears to be
30             stable compared to the values recorded for the
31             entire region"
32         elif(flag == 3):
33             str = str + " This PM10 value turns out
34             to be closer to the maximum value recorded
35             in the entire region"
36         elif(flag == 4):

```

```

37         str = str + " This PM10 value also turns out
38         to be a new maximum for the entire region"
39         str=str+"
40     return str
41
42 def freq(flag,season):
43     str=''
44     if(not np.isnan(flag)):
45         if(flag == 0):
46             str = str + " and it is infrequent in {}".format(season)
47             #0-29%
48         elif(flag == 1):
49             str = str + " and it is quite frequent in
50             {}".format(season) #30-59%
51         elif(flag == 2):
52             str = str + " and it is very frequent in
53             {}".format(season) #60-100%
54         str=str+"
55     return str

```

Come prima cosa si è creati un file di supporto per la conversione. In questo script, sono state definite 3 funzioni relative a 4 campi del dataset con dati strutturati, specificamente ai campi PM10, MINMAX, FREQ, SEASON. Le tre funzioni sono:

- `limitvalue(pm10)`: tale funzione prende in input un pm10 e restituisce un output in formato stringa, tale che faccia capire se il pm10 registrato supera o rientra nei valori limite. Sui valori limite ci si è basati su quelli definiti dall'Unione europea.
- `minmax(flag)`: tale funzione prende in input una flag relativa al campo MINMAX (valori da 0-4) e restituisce un output in formato stringa, tale che faccia capire se il pm10 registrato risulta essere un nuovo minimo, massimo per l'intera regione oppure se è vicino, stabile, lontano dai minimi e massimi.
- `freq(flag,season)`: tale funzione prende in input una flag e una season, relative rispettivamente al campo FREQ e SEASON. La funzione restituisce in output una stringa, tale che faccia capire quanto è frequente quel pm10 registrato, durante la stagione di appartenenza.

```
1 from tkinter.messagebox import NO
2 from scalingSupport import *
3 import pandas as pd
4 import numpy as np
5 import os
6 #creo frasi partendo dal csv verso un txt
7
8 dataset = pd.read_csv("dataset.csv")
9
10 datasetTesto = open('dataset.txt','a',encoding='utf8')
11
12 for index, row in dataset.iterrows():
13     PROVINCIA = row['PROVINCIA']
14     REGIONE = row['REGIONE']
15     PM10 = row['PM10']
16     MINMAX = row['MINMAX']
17     SEASON = row['SEASON']
18     FREQ = row['FREQ']
19
20     str1 = " At {} province of {}, has been recorded"
21     .format(PROVINCIA,REGIONE)
22     str2 = limitvalue(PM10)
23     str3 = minmax(MINMAX)
24     str4 = freq(FREQ,SEASON)
25
26     datasetTesto.write('<|stroftext|>;{};{};PM10 {}
27 ;MINMAX {};{};FREQ {}: '
28 .format(PROVINCIA,REGIONE,PM10,MINMAX,SEASON,FREQ)+
29     str1+
30     str2+'.'+
31     str3+
32     str4+'.<|endoftext|>\n'
33 )
34
35 datasetTesto.close()
```

Tale codice, rappresenta lo script principale che dovrà essere eseguito per effettuare la conversione. Si parte da un dataset in formato csv e si ottiene un dataset finale in formato txt, dove ogni riga rappresenta una frase (istanza). Questo script fa uso delle funzioni di supporto citate prima, ed inoltre serve per costruire lo scheletro generale dell'intera frase. La frase è stata costruita attraverso il seguente pattern ideato:

- 1) La frase inizierà sempre con "<|stroftext|>", ciò serve per far capire al modello che dopo questa parola speciale, inizia il contenuto rilevante.
- 2) Subito dopo la parola speciale esso segue ";{};{};PM10 {} ;MINMAX {};{};FREQ {}", dove ogni "{}" rappresenta un valore. Nel primo sarà posto la provincia, nel secondo la regione, infine nei seguenti, vengono posti valori di cui il significato risulta essere esplicito. Quest'intera stringa rappresenta l'input su cui il modello dovrà generare la frase.
- 3) Di conseguenza verrà concatenata alla stringa precedente la seguente stringa ": ". Tale stringa farà da separatore, tra la stringa di input e la stringa di output, utile proprio al modello per comprendere chi è l'input e chi è l'output.
- 4) Ad essa verrà concatenata la frase costruita attraverso le precedenti funzioni di supporto, in modo da rappresentare l'output tale che il modello dovrà generare.
- 5) Infine la frase si conclude sempre con "<|endoftext|>\n", per indicare che la frase è finita e che nella riga successiva ci potrà essere una nuova frase.

Alcuni esempi di frasi interne al dataset finale sono:

"<|stroftext|>;Sondrio;Lombardia;PM10 -5.6;MINMAX 2;spring;FREQ 2: At Sondrio province of Lombardia, has been recorded a PM10 equal to -5.6 and it exceeds the limit value. This PM10 value appears to be stable compared to the values recorded for the entire region and it is very frequent in spring.<|endoftext|>"

"<|stroftext|>;Bolzano;Trentino-Alto Adige;PM10 9.2;MINMAX 0;autumn;FREQ 0: At Bolzano province of Trentino-Alto Adige, has been recorded a PM10 equal to 9.2 and it is within the limit value. This PM10 value also turns out to be a new minimum for the entire region and it is infrequent in autumn.<|endoftext|>"

"<|stroftext|>;Ancona;Marche;PM10 0.0;MINMAX 0;winter;FREQ 1: At Ancona province of Marche, has been recorded a PM10 equal to 0.0 and it is within the limit value. This PM10 value also turns out to be a new minimum for the entire region and it is quite frequent in winter.<|endoftext|>"

"<|stroftext|>;Salerno;Campania;PM10 15.0;MINMAX 1;spring;FREQ 1: At Salerno province of Campania, has been recorded a PM10 equal to 15.0 and it is within the limit value. This PM10 value turns out to be closer to the lowest value recorded in the entire region and it is quite frequent in spring.<|endoftext|>"

La dimensione del dataset finale, risulta essere di 573.155KB e sono state create delle frasi in lingua inglese, per ottenere migliori performance da parte del modello.

3.4 Addestramento

Con l'avvenuta installazione del modello 774M e la creazione del dataset finale, si procede con il training del modello.

```
1 import gpt_2_simple as gpt2
2 import os
3 os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
4 #nasconde i drive cuda a tensorflow per far si che
5 #venga utilizzata la cpu e non la gpu
6
7 sess = gpt2.start_tf_sess(threads=48)
8 #inizio sessione tensorflow creata gpt-2-simple
9
10
11
12 file_name = "Dataset\\dataset.txt" #assegnazione file
13
14 gpt2.finetune(sess,
15               dataset=file_name,
16               model_name='774M',
17               steps=300,
18               sample_every=8000, #per forzare il sample ogni 8000
19               #cosi da non farlo stampare, altrimenti va in errore
20               #con il modello 774
21               restore_from='fresh'
22               )
23
24 #finetune
```

GPT-2-SIMPLE per il training offre la possibilità di poter settare dei parametri. I parametri settati sono:

- `model_name`: indica il nome del modello che vogliamo utilizzare, 774M nel nostro caso.
- `steps`: indica il numero di iterate di addestramento, 300 nel nostro caso.

- `sample_every`: indica ogni qual volta il modello debba generare degli esempi, durante l'addestramento. È un parametro che nella versione 774M provoca delle eccezioni, quindi, nel nostro caso è stato impostato ad 8000, per forzare la non generazione di esempi.
- `restore_from`: indica il checkpoint da cui si vuol partire per il fine-tuning. Visto che è il primo addestramento del modello, abbiamo settato il valore "fresh". Se non fosse stato così, allora era possibile utilizzare il valore "latest", che permette di partire da un checkpoint salvato.

La sessione di addestramento del modello è durata circa 3 giorni, ottenendo una loss media di circa 0.03. Finito l'addestramento, GPT-2-SIMPLE salverà il tutto in una cartella chiamata checkpoint.

3.5 Funzione di generazione

Sebbene il modello sia stato addestrato, ricordiamo che l'intero sistema è costituito da due layer, quindi, non possiamo subito passare all'esecuzione della generazione. In questo capitolo vedremo la costruzione di uno script in cui viene definita una funzione di generazione che verrà utilizzata da uno script main alla fine della realizzazione dei due layer.

```
1 import gpt_2_simple as gpt2
2 import os
3 from googletrans import Translator
4
5 #inizializzo i parametri per la generazione
6 #l'inizializzazione è importante
7 #inserirli al di fuori della funzione per far sì che venga
8 #eseguita 1 sola volta ed evitare di creare nuovi sessioni
9 #e caricare sempre lo stesso checkpoint
10
11 traduttore = Translator()
12 os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
13 sess = gpt2.start_tf_sess()
```

```

14 gpt2.load_gpt2(sess, run_name='run1',
15 checkpoint_dir="GptLayer/Training/checkpoint")
16
17 #genero e traduco la frase
18 def generate(prompt):
19     fraseIngl = gpt2.generate(sess,
20     checkpoint_dir="GptLayer/Training/checkpoint", run_name='run1',
21     temperature=0.1, prefix=prompt, truncate="<|endoftext|>",
22     return_as_list=True)[0].replace(prompt, '')
23     return ((traduttore.translate(fraseIngl, src="en", dest="it")).text)

```

Tale script, una volta importato dallo script main, farà in modo che verrà inizializzata la sessione e il caricamento del modello come primo step. Inoltre è stata definita la funzione di generazione, chiamata: "generate(prompt)". Tale funzione prende in input una stringa e restituisce in output una stringa generata dal modello. L'output, prima di essere restituito, verrà processato per evitare caratteri, parole "rumorose" e che siano in lingua italiana. Proprio perché abbiamo bisogno di un output strettamente relato a dati ambientali reali, il prompt prima di essere passato alla funzione, sarà processato a sua volta dal layer di analisi. In conclusione con la funzione di generazione, è stata così terminata la realizzazione del primo layer, dedicato esclusivamente all'addestramento e al futuro uso del modello.

3.6 Pre-processing dell'input

Dopo la costruzione del layer dedicato al modello, si procede con la realizzazione del layer di analisi, con l'obiettivo di pre-processare l'input prima che venga dato in pasto alla funzione di generazione. Grazie al pre-processing, si otterrà un input strettamente relato a dati ambientali aggiornati per ogni quadrimestre. Attraverso quest'input, il modello svolgerà la funzione di "classificatore", classificando il significato di ogni singolo dato, generando poi un'intera frase.

Come prima cosa, per la realizzazione di questo layer, abbiamo bisogno di dati annuali, che dovranno poi essere aggiornati ad ogni quadrimestre. Si è fatta una raccolta annuale dei dati di ogni regione dell'Italia (21 per l'esattezza e forniti da Sense Square), partendo dal 30-11-2021 fino al 30-11-2022. Ogni file di dato, sarà in formato csv e rinominato con l'esatto nome della regione così gestita da Sense Square.

Abruzzo.csv	16/01/2023 19:09	File di origine Co...	22 KB
Basilicata.csv	16/01/2023 19:09	File di origine Co...	26 KB
Calabria.csv	16/01/2023 19:09	File di origine Co...	42 KB
Campania.csv	16/01/2023 19:09	File di origine Co...	47 KB
Emilia-Romagna.csv	16/01/2023 19:09	File di origine Co...	30 KB
Friuli-Venezia Giulia.csv	16/01/2023 19:09	File di origine Co...	28 KB
Lazio.csv	16/01/2023 19:09	File di origine Co...	29 KB
Liguria.csv	16/01/2023 19:09	File di origine Co...	27 KB
Lombardia.csv	16/01/2023 19:09	File di origine Co...	44 KB
Marche.csv	16/01/2023 19:09	File di origine Co...	27 KB
Molise.csv	16/01/2023 19:09	File di origine Co...	24 KB

Figura 3.3: Visualizzazione parziale dei dati raccolti.

Tutti i dati sono composti da un campo interno chiamato “timestamp”. Tale campo, salva la data giornaliera della registrazione degli inquinanti per la specifica regione. Si è realizzati il seguente script, che ha lo scopo di rimpiazzare la data con la stagione di appartenenza, a causa del fatto che l'analisi viene eseguita per stagione.

```

1 from datetime import date, datetime
2 import pandas as pd
3 import numpy as np
4 import os
5
6 #rimpiazzo tutte le date nella colonna timestamp di ogni dataset,
7 #con le stagioni appartenenti
8
9 def get_season(now):
10     Y = 2000 # dummy leap year to allow input X-02-29 (leap day)

```

```

11     #serve per cambiare dinamicamente le date con le stagioni
12     seasons = [('winter', (date(Y, 1, 1), date(Y, 3, 20))),
13                ('spring', (date(Y, 3, 21), date(Y, 6, 21))),
14                ('summer', (date(Y, 6, 22), date(Y, 9, 22))),
15                ('autumn', (date(Y, 9, 23), date(Y, 12, 22))),
16                ('winter', (date(Y, 12, 23), date(Y, 12, 31)))]
17
18     if isinstance(now, datetime):
19         now = now.date()
20     now = now.replace(year=Y)
21     return next(season for season, (start, end) in seasons
22                if start <= now <= end)
23
24 provreg = pd.read_csv("provreg.csv")
25
26 for indice, riga in provreg.iterrows():
27     dataset = pd.read_csv(riga['REGIONE']+'.csv')
28     for index, row in dataset.iterrows():
29         dataset.at[index, 'timestamp'] =
30             get_season(datetime.strptime(row['timestamp'], '%Y-%m-%d'))
31     dataset.to_csv(riga['REGIONE']+'.csv', index=False, encoding='utf-8')

```

Dopo la raccolta dati, si è proceduti con la costruzione di due script, dove vengono definiti nel loro interno delle funzioni di analisi e di supporto, che verranno richiamate dallo script main per il pre-processing dell'input.

```

1 from datetime import date, datetime
2 import pandas as pd
3 import numpy as np
4 import os
5
6 Y = 2000 # dummy leap year to allow input X-02-29 (leap day)
7 #serve per cambiare dinamicamente le date con le stagioni
8 seasons = [('winter', (date(Y, 1, 1), date(Y, 3, 20))),
9            ('spring', (date(Y, 3, 21), date(Y, 6, 21))),
10            ('summer', (date(Y, 6, 22), date(Y, 9, 22))),
11            ('autumn', (date(Y, 9, 23), date(Y, 12, 22))),
12            ('winter', (date(Y, 12, 23), date(Y, 12, 31)))]
13
14 #limite del pm10
15 pm_limite=50
16

```

```

17 #funzione che normalizza il valore limite
18 def limitvalue(pm10):
19     return round(pm_limite - pm10,1)
20
21 #funzione che mi calcola se si tratta:
22 #di un nuovo minimo - 0
23 #di un valore vicino al minimo - 1
24 #di un valore stabile - 2
25 #di un valore vicino al massimo - 3
26 #di un nuovo massimo - 4
27 def minmax(regione,pm10):
28     dataset = pd.read_csv("AnalisysLayer/dati/"+regione+".csv")
29     min = round((pm_limite-float(dataset.min()['pm10'])), 1)
30     max = round((pm_limite-float(dataset.max()['pm10'])), 1)
31     avg = round(((min+max)/2), 1)
32     avgmin = round(((min+avg)/2), 1)
33     avgmax = round(((avg+max)/2), 1)
34     if pm10>min :
35         offset = 0
36     elif (pm10<=min) and (pm10>avgmin):
37         offset = 1
38     elif (pm10<=avgmin) and (pm10>=avgmax):
39         offset = 2
40     elif (pm10<avgmax) and (pm10>=max):
41         offset = 3
42     elif pm10<max:
43         offset = 4
44     return offset
45
46 #funzione che restituisce la stagione a cui appartiene la data
47 #che è stata presa in input
48 def get_season(now):
49     if isinstance(now, datetime):
50         now = now.date()
51     now = now.replace(year=Y)
52     return next(season for season, (start, end) in seasons
53                 if start <= now <= end)
54
55 #calcolo della frequenza di quel pm10 sulla base
56 #della stagione e della regione
57 #per calcolare la frequenza sono stati definite
58 #classi di equivalenza di range 10
59 #restituisce 0 se è poco frequente 0-29%,
60 #1 se è abbastanza frequente 30-59%,
61 #2 se è molto frequente 60-100%
62 def freq(regione,stagione,pm10):
63     dataset = pd.read_csv("AnalisysLayer/dati/"+regione+".csv")

```

```

64     size = 1 # va contato lo stesso numero pm10 di input perchè
65     #e come se fosse registrato
66     #creo la classe di equivalenza di range 10
67     rangemin = int(int(int(pm10)/10)*10)
68     rangemax = rangemin + 9
69     count = 1 #perchè quel numero già esiste in quel range
70     #quindi va contato
71     for index, row in dataset.iterrows():
72         if (row['regione'] == regione) and (row['timestamp'] == stagione):
73             if(not np.isnan(row['pm10'])):
74                 size=size+1
75                 if(row["pm10"]>=rangemin) and (row["pm10"]<=rangemax):
76                     count = count + 1
77     frequenza = ((count * 100)/size)
78     if frequenza < 30:
79         return 0
80     elif frequenza < 60:
81         return 1
82     elif frequenza < 100:
83         return 2

```

Nel seguente script, sono state definite 4 funzioni di analisi, che verranno utilizzate dallo script di pre-processing. Le funzioni sono:

- `limitvalue(pm10)`: tale funzione prende in input un pm10 in formato float e restituisce lo stesso valore, normalizzato con il valore limite del pm10. Se tale valore risulta essere < 0 , allora vuol dire che supera il valore limite, altrimenti, si tratta di un valore che rientra nei limiti.
- `minmax(regione,pm10)`: tale funzione prende in input un pm10 ed una regione e restituisce un valore tra 0 e 4. Si controlla nel set di dati di una specifica regione, chi è il massimo e il minimo valore di pm10. Attraverso queste informazioni, si restituisce il valore che indica la distanza tra il pm10 (preso in input dalla funzione) con il minimo e il massimo.
- `get_season(now)`: tale funzione prende in input una data in formato datetime e restituisce la stagione di appartenenza.

- `freq(regione,stagione,pm10)`: tale funzione prende in input una regione, una stagione ed un pm10 e restituisce la frequenza di quel pm10 per la specifica regione in una data stagione, attraverso un valore che può andare da 0 a 2. I dati vengono clusterizzati in classi di equivalenza di distanza 10, in modo da non avere troppa diversificazione.

```

1  import pandas as pd
2  import numpy as np
3  import AnalisisLayer.analisi as analisi
4  import os
5
6  #dati gli input originali creo una stringa formattata on un certo modo
7  #per far si che gpt2 capisca l'input
8  def process(provincia,regione,pm10,data):
9      #ottengo il pm10 normalizzato
10     scalpm10 = analisi.limitvalue(pm10)
11     #ottengo il minmax
12     minmax = analisi.minmax(regione,scalpm10)
13     #ottengo la stagione associata a quella data
14     stagione = analisi.get_season(data)
15     #ottengo la frequenza di quel pm10 sulla base
16     #della regione e della stagione
17     freq = analisi.freq(regione,stagione,pm10)
18     #definisco la stringa
19     str= "<|stroftext|>;{};{};PM10 {};MINMAX {};{};FREQ {}:"
20     .format(provincia,regione,scalpm10,minmax,stagione,freq)
21     return str,scalpm10,minmax,stagione,freq

```

Tale codice è lo script di pre-processing, utilizzato dallo script main, per ottenere l'input pre-processato. Il seguente script fa uso delle funzioni di analisi per ottenere un input costantemente aggiornato con i dati ambientali regionali. Infine si conclude così la realizzazione del layer di analisi.

3.7 Messa in produzione

Conclusa la realizzazione dei due layer, si è passati alla costruzione di uno script main, necessario per la messa in produzione dell'intero sistema.

```

1  import os
2
3  os.chdir('C:\\Users\\SSQAI\\Desktop\\PELUSO\\gpt2\\SENSEAI')
4
5  import AnalisisLayer.processing as processing
6  import AnalisisLayer.analisi as analisi
7  import GptLayer.generazione as generatore
8  from datetime import datetime
9  import json
10 import requests
11 import concurrent.futures
12 from time import time, sleep
13
14 lista = []
15 #Ottengo tutte le regioni
16 urlRegioni="https://square.sensesquare.eu:5001//elenco_regioni"
17
18 paramRegioni={
19     'naz_name': 'Italia'
20 }
21
22 responseRegioni=requests.post(urlRegioni,data=paramRegioni)
23
24 dictRegioni=json.loads(responseRegioni.text)
25
26 for reg in dictRegioni["result"]:
27     #ottengo tutte le province con il loro pm10 giornaliero
28     url="https://square.sensesquare.eu:5001//placeView"
29     todayData=datetime.now()
30     strData=str(str(todayData.year) + "-0" + str(todayData.month) +
31               "-" + str(todayData.day))
32
33     paramProvince={
34         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
35         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
36         'NON E POSSIBILE VISUALIZZAREi': 'NON E POSSIBILE VISUALIZZARE',
37         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
38         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
39         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',

```

```

40     'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZAR',
41     'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE'
42 }
43
44 responseProvince=requests.post(url,data=paramProvince)
45 dictProvince=json.loads(responseProvince.text)
46
47 for prov in dictProvince['result']:
48     #mi prendo solo cio che mi serve
49     regione = reg
50     provincia = prov
51     pm10 = float(dictProvince["result"][prov]["pm10"])
52     data = datetime.strptime(strData, '%Y-%m-%d')
53     #ottengo l'input processato per gpt2
54     prompt, scalpm10,minmax,stagione,freq =
55     processing.process(provincia,regione,pm10,data)
56     #genero la frase
57     frase = generatore.generate(prompt)
58     #ottenendo la frase da gpt2 devo cambiare
59     #il valore di pm10 normalizzato, rimpiazzandolo
60     str11="{}".format(scalpm10)).replace(".",",")
61     str12="{}".format(scalpm10))
62     str2="{}".format(pm10)).replace(".",",")
63     frase = frase.replace(str11,str2)
64     frase = frase.replace(str12,str2)
65     #creo stringa json
66     lista.append({ 'regione' : reg, 'provincia' : prov,
67     'pm10' : pm10, 'data' : strData, 'frase' : frase })
68
69 strutturajson = {
70     'frasi' : lista
71 }
72 stringajson = json.dumps(strutturajson)
73 #stampo la frase
74 with open("frasiGiornaliere/"+strData+".json", "a") as f:
75     f.write(stringajson)

```

Tale script si pone l'obiettivo di ottenere dal server aziendale, il nome della provincia, della regione e il rispettivo pm10 registrato in quel dato istante, salvando anche la data. Successivamente richiama il layer di analisi per il pre-processing dell'input, ottimale per la generazione. Data la stringa di input pre-processata, subito sarà passata alla funzione di generazione, ottenendo una frase strettamente legata ai dati ambientali, in lingua italiana. Lo script genererà 107 frasi e solo alla fine dell'ultima generazione, salverà tutte le frasi in un file JSON rinominato con il nome della data, all'interno di una cartella dedicata. Infine è stata programmata come attività interna al computer aziendale, l'esecuzione di tale script ogni giorno alle ore 12:00, ciò vuol dire che l'intero sistema genererà un file al giorno.

```
{
  "frasi": [
    {
      "regione": "Piemonte",
      "provincia": "Biella",
      "pm10": 30.2,
      "data": "2023-02-25",
      "frase": "A Biella provincia del Piemonte, \u00e8 stato registrato un PM10 pari a 30,2 ed \u00e8 entro il valore limite. Qu"
    },
    {
      "regione": "Piemonte",
      "provincia": "Novara",
      "pm10": 38.48,
      "data": "2023-02-25",
      "frase": "A Novara provincia di Piemonte, \u00e8 stato registrato un PM10 pari a 38,48 ed \u00e8 entro il valore limite. Qu"
    },
    {
      "regione": "Piemonte",
      "provincia": "Cuneo",
      "pm10": 31.08,
      "data": "2023-02-25",
      "frase": "A Cuneo, in provincia di Piemonte, \u00e8 stato registrato un PM10 pari a 31,08 e rientra nel valore limite. Qu"
    },
    {
      "regione": "Piemonte",
      "provincia": "Torino",
      "pm10": 26.22,
      "data": "2023-02-25",
      "frase": "A Torino provincia del Piemonte, \u00e8 stato registrato un PM10 pari a 26,22 ed \u00e8 entro il valore limite. Qu"
    },
    {
      "regione": "Piemonte",
      "provincia": "Asti",
      "pm10": 33.79,
      "data": "2023-02-25",
      "frase": "Ad Asti provincia del Piemonte, \u00e8 stato registrato un PM10 pari a 33,79 ed \u00e8 entro il valore limite. Qu"
    },
    {
      "regione": "Piemonte",
      "provincia": "Verbano-Cusio-Ossola",
      "pm10": 14.59,
      "data": "2023-02-25",
      "frase": "Al Verbano-Cusio-Ossola provincia piemontese, \u00e8 stato registrato un PM10 pari a 14,59 ed \u00e8 entro il v"
    }
  ]
}
```

Figura 3.4: Visualizzazione parziale del file delle frasi generate.

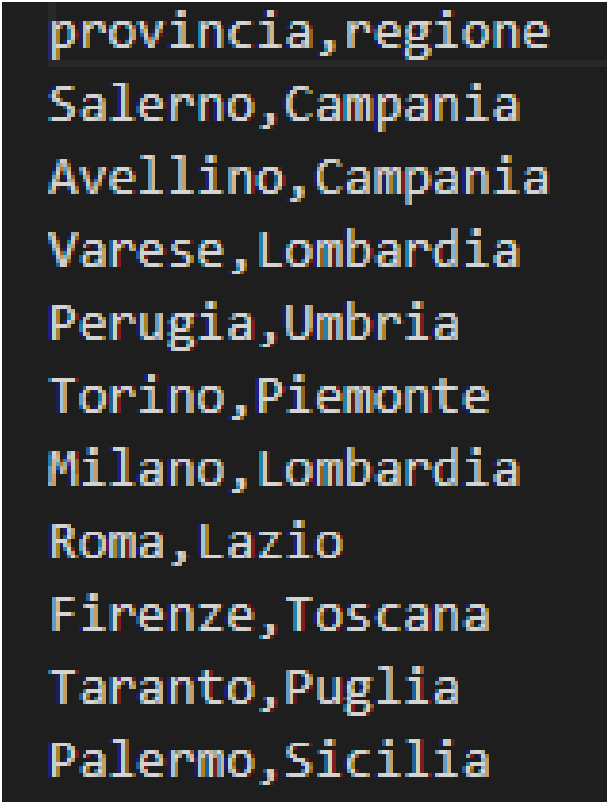
CAPITOLO 4

Valutazione

In conclusione con l'implementazione dell'intero sistema (Capitolo 3), è stato eseguito successivamente un'operazione molto importante, riguardante la progettazione di un modello, che è la sua valutazione. Il modello è stato valutato in maniera empirica, non utilizzando metriche. La valutazione consiste solo nel generare 10 frasi in orari differenti, per poi analizzarle "a mano". La motivazione della scelta di questa tipologia di testing, è data dal fatto che le frasi debbano essere veritiere con i dati registrati in un dato momento. Per eseguire il testing è stato creato uno script di test per la generazione delle 10 frasi (conservandole in un file JSONL), per poi essere valutate alla fine.

4.1 Script di test

Per prima cosa sono state scelte in modo del tutto casuale, 10 province appartenenti rispettivamente a 10 regioni differenti tra loro. Tali province e regioni sono state inserite all'interno di un file csv.



```
provincia, regione
Salerno, Campania
Avellino, Campania
Varese, Lombardia
Perugia, Umbria
Torino, Piemonte
Milano, Lombardia
Roma, Lazio
Firenze, Toscana
Taranto, Puglia
Palermo, Sicilia
```

Figura 4.1: Province-regioni scelte casualmente.

Subito dopo, si è passati alla creazione dello script di test, per la generazione delle 10 frasi da valutare:

```
1 import AnalisysLayer.processing as processing
2 import AnalisysLayer.analisi as analisi
3 import GptLayer.generazione as generatore
4 from time import time, sleep
5 import pandas as pd
6 import datetime
7 import requests
8 import json
9
10 url="https://square.sensesquare.eu:5001//placeView"
11
12 #setto le province e regioni da testare
13 provreg = pd.read_csv('test.csv')
14 #apro il file dei risultati
15 risultati = open('risultati.jsonl', 'a', encoding='utf8')
16
17 for index, row in provreg.iterrows():
```

```

18     provi = row["provincia"]
19     reg = row["regione"]
20     #setto i parametri per la chiamata POST
21     todayDate=datetime.datetime.now()
22     strData=str(str(todayDate.year) + "-0"+ str(todayDate.month)+
23     "-" + str(todayDate.day))
24     strHours=str(str(todayDate.hour)+":"+str(todayDate.minute))
25     print(provi,reg)
26     param={
27         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
28         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
29         'NON E POSSIBILE VISUALIZZAREi': 'NON E POSSIBILE VISUALIZZARE',
30         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
31         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
32         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE',
33         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZAR',
34         'NON E POSSIBILE VISUALIZZARE': 'NON E POSSIBILE VISUALIZZARE'
35     }
36
37     #faccio la chiamata ed ottengo i dati
38     responseProvince=requests.post(url,data=param)
39     dictProvince=json.loads(responseProvince.text)
40     for prov in dictProvince["result"]:
41         #trovo la provincia che mi interessa
42         if prov == provi:
43             #genero la frase
44             data = todayDate
45             provincia = provi
46             regione = reg
47             pm10 = float(dictProvince["result"][prov]['pm10'])
48             prompt = processing.process(provincia,regione,pm10,data)
49             print("Provincia: {} - Regione: {} - PM10: {} - DATA: {}".
50             .format(provincia,regione,pm10,data))
51             print("il prompt: "+prompt)
52             frase = generatore.generate(prompt)
53             str1("{}".format(analisi.limitvalue(pm10))).replace(".",",")
54             str2("{}".format(pm10)).replace(".",",")
55             frase = frase.replace(str1,str2)
56             #mi creo il risultato e lo scrivo sul file json
57             data = {
58                 'input' : {'provincia' : provincia, 'regione' : regione,
59                 'pm10' : '{}'.format(pm10), 'data' : str(todayDate),
60                 'prompt' : prompt},
61                 'output' : frase
62             }
63             stringajson = json.dumps(data)
64             risultati.write(stringajson+"\n")

```

```
65         #mi fermo per 3 ore
66         print("Risultato scritto")
67         sleep(10800)
68
69     risultati.close()
```

Tale script per ogni provincia, otterrà dal server di Sense Square, il pm10 registrato in quel dato momento. Tramite queste informazioni, il sistema genererà una frase che verrà salvata in formato JSON in un file JSONL. Visto che ciò si ripeterà per le 10 province scelte, si avrà un file JSONL contenente le 10 frasi in formato JSON. Le 10 frasi generate all'interno del file JSONL, risultano essere:

```
""input": "provincia": "Salerno", "regione": "Campania", "pm10": "10.77", "data": "2023-01-17 18:28:56.796814", "prompt": "<| stroftext |>;Salerno;Campania;PM10 39.2;MIN-MAX 1;winter;FREQ 1: ", "output": "A Salerno provincia della Campania,    stato registrato un PM10 pari a 10,77 ed    entro il valore limite. Questo valore di PM10 risulta essere pi   vicino al valore pi   basso registrato nell'intera regione ed    abbastanza frequente in inverno.""
```

```
""input": "provincia": "Avellino", "regione": "Campania", "pm10": "6.18", "data": "2023-01-17 21:40:32.294003", "prompt": "<| stroftext |>;Avellino;Campania;PM10 43.8;MIN-MAX 1;winter;FREQ 0: ", "output": "Ad Avellino provincia di Campania,    stato registrato un PM10 pari a 6,18 ed    entro il valore limite. Questo valore di PM10 risulta essere pi   vicino al valore pi   basso registrato nell'intera regione ed    poco frequente in inverno.""
```

```
""input": "provincia": "Varese", "regione": "Lombardia", "pm10": "24.8", "data": "2023-01-18 00:52:07.333518", "prompt": "<| stroftext |>;Varese;Lombardia;PM10 25.2;MIN-MAX 2;winter;FREQ 0: ", "output": "A Varese provincia di Lombardia,    stato registrato un PM10 pari a 24,8 ed    entro il valore limite. Questo valore di PM10 appare stabile rispetto ai valori registrati per l'intera regione ed    poco frequente nel periodo invernale.""
```

“input": "provincia": "Perugia", "regione": "Umbria", "pm10": "3.73", "data": "2023-01-18 04:03:53.868520", "prompt": "<| strof text |>;Perugia;Umbria;PM10 46.3;MINMAX 1;winter;FREQ 0: ", "output": "A Perugia provincia dell'Umbria, è stato registrato un PM10 pari a 3,73 ed è entro il valore limite. Questo valore di PM10 risulta essere più vicino al valore basso registrato nell'intera regione ed è poco frequente in inverno.”

“input": "provincia": "Torino", "regione": "Piemonte", "pm10": "25.5", "data": "2023-01-18 07:15:42.122707", "prompt": "<| strof text |>;Torino;Piemonte;PM10 24.5;MINMAX 1;winter;FREQ 0: ", "output": "A Torino provincia del Piemonte, è stato registrato un PM10 pari a 25,5 ed è entro il valore limite. Questo valore di PM10 risulta essere più vicino al valore basso registrato nell'intera regione ed è poco frequente in inverno.”

“input": "provincia": "Milano", "regione": "Lombardia", "pm10": "41.04", "data": "2023-01-18 10:27:31.816945", "prompt": "<| strof text |>;Milano;Lombardia;PM10 9.0;MINMAX 2;winter;FREQ 0: ", "output": "A Milano provincia Lombardia, è stato registrato un PM10 pari a 9.0 ed è entro il valore limite. Questo valore di PM10 appare stabile rispetto ai valori registrati per l'intera regione ed è poco frequente nel periodo invernale.”

“input": "provincia": "Roma", "regione": "Lazio", "pm10": "9.86", "data": "2023-01-18 13:39:30.069866", "prompt": "<| strof text |>;Roma;Lazio;PM10 40.1;MINMAX 1;winter;FREQ 0: ", "output": "A Roma provincia del Lazio, è stato registrato un PM10 pari a 9,86 e si rientra nel valore limite. Questo valore di PM10 risulta essere più vicino al valore basso registrato nell'intera regione ed è poco frequente in inverno.”

“input": "provincia": "Firenze", "regione": "Toscana", "pm10": "9.34", "data": "2023-01-18 16:51:31.975672", "prompt": "<| strof text |>;Firenze;Toscana;PM10 40.7;MINMAX 1;winter;FREQ 0: ", "output": "A Firenze provincia Toscana, è stato registrato

un PM10 pari a 9,34 ed 00e8 entro il valore limite. Questo valore di PM10 risulta essere pi00f9 vicino al valore pi00f9 basso registrato nell'intera regione ed 00e8 poco frequente in inverno.""

```
""input": "provincia": "Taranto", "regione": "Puglia", "pm10": "8.88", "data": "2023-01-18 20:03:39.241989", "prompt": "<| stroftext |>;Taranto;Puglia;PM10 41.1;MINMAX 1;winter;FREQ 0: ", "output": "A Taranto provincia di Puglia, 00e8 stato registrato un PM10 pari a 8,88 e si rientra nel valore limite. Questo valore di PM10 risulta essere pi00f9 vicino al valore pi00f9 basso registrato nell'intera regione ed 00e8 poco frequente in inverno.""
```

```
""input": "provincia": "Palermo", "regione": "Sicilia", "pm10": "16.72", "data": "2023-01-18 23:15:52.678982", "prompt": "<| stroftext |>;Palermo;Sicilia;PM10 33.3;MINMAX 1;winter;FREQ 2: ", "output": "A Palermo provincia di Sicilia, 00e8 stato registrato un PM10 pari a 16,72 ed 00e8 entro il valore limite. Questo valore di PM10 risulta essere pi00f9 vicino al valore pi00f9 basso registrato nell'intera regione ed 00e8 molto frequente in inverno.""
```

Per ogni frase, si possono notare come vengono stampati dei caratteri speciali. Questo è dato dal salvataggio delle frasi in un file, infatti ciò non andrà ad intaccare la valutazione.

4.2 Risultati

Dopo la generazione, le frasi interne al file JSONL, sono state valutate “a mano”, poiché, si deve verificare se esse siano conformi ai dati su cui si sono fatte le analisi di pre-processing dell’input. Per ogni frase si è controllati se essa rispecchi l’input pre-processato, infatti, è stato salvato come ulteriore informazione all’interno di ogni frase nel JSONL, proprio per consentire un confronto e non perdere l’informazione. Infine per ogni provincia su cui è stata generata la frase, si è controllati nel set di dati necessario per il pre-processing, se tale frase sia veritiera, ovvero, se ad esempio si parla realmente di un nuovo minimo o nuovo massimo, oppure, se tale pm10 per quella regione è realmente frequente in una data stagione. I risultati si sono rivelati molto soddisfacenti, poiché le frasi erano conformi al 100% con l’output atteso, soprattutto perché la frase assume il significato di un report che si rifà a dati reali. L’unico punto su cui si potrebbe discutere, è sulla dinamicità delle frasi. Le frasi risultano essere molto statiche e variano solo al variare dei dati di input.

Conclusioni e sviluppi futuri

L'utilizzo del sistema implementato, consentirebbe di fornire in maniera più semplice delle indicazioni verso l'utenza di Sense Square. Il tutto poiché si dà un'interpretazione ai dati, attraverso delle frasi generate dal modello. Si è partiti da varie documentazioni per una comprensione del dominio applicativo e del dominio delle soluzioni, ciò inoltre, è servito anche per la costruzione del secondo capitolo di background. Grazie allo studio iniziale e anche alla comprensione degli obiettivi dall'azienda, si è fatta una scelta delle tecnologie che fossero adatte per essa, come ad esempio l'uso di GPT-2 e non di GPT-3, a causa del fatto che l'azienda aveva bisogno di un sistema proprietario e compatibile con l'hardware a disposizione. Subito dopo si è proseguiti con la costruzione dell'intero sistema, dove se ne parla nel terzo capitolo. La difficoltà maggiore, è stata proprio il creare un dataset iniziale, perché si doveva capire quali dati fossero adatti per il problema in questione (essendo molto specifico e non avendo dati a disposizione per esso). Grazie alla valutazione del modello, dove se ne parla nel quarto capitolo, si è capiti che esso fosse adatto per la piattaforma. Esso non genera alcun errore ed ogni frase soddisfa gli obiettivi imposti dall'azienda. Le frasi risultano essere molto statiche e variano al variare dei dati, a causa della relazione stretta con i dati, infatti con ciò, si è analizzati quali fossero i possibili sviluppi futuri. Si è pensati a diversi possibili scenari, come ad esempio:

automatizzare l'aggiornamento quadrimestrale dei dati per il pre-processing dell'input; effettuare un ulteriore addestramento del modello, con frasi diverse ma che assumono lo stesso significato delle frasi dell'attuale dataset, oppure, migliorare ulteriormente il layer di analisi per il pre-processing, per ottenere delle analisi più accurate, attraverso anche l'implementazione di un altro modello di deep learning dedicato.

Bibliografia

- [1] “Immagini della piattaforma sensesquare,” 2023, controllata il 28-02-2023. [Online]. Available: <https://square.sensesquare.eu/> (Citato alle pagine iii, 2 e 3)
- [2] Datecon, “Digital voice assistant for data retrieval in reporting,” 2022, controllata il 28-02-2023. [Online]. Available: <https://www.datecon.com/en/journal/digital-voice-assistant-data-retrieval-reporting> (Citato alle pagine iii e 7)
- [3] SAP, “Che cos’è il machine learning?” controllata il 28-02-2023. [Online]. Available: <https://www.sap.com/italy/insights/what-is-machine-learning.html> (Citato alle pagine iii e 11)
- [4] Appuntioss, “Sistema nervoso - i neuroni,” controllata il 28-02-2023. [Online]. Available: <https://www.appuntioss.it/anatomia-corpo-sistema-nervoso-i-neuroni/> (Citato alle pagine iii e 12)
- [5] C. D. Nardo, “Cos’è l’intelligenza artificiale?” 2021, controllata il 28-02-2023. [Online]. Available: <https://deltalogix.blog/2021/03/10/intelligenza-artificiale/> (Citato alle pagine iii e 13)
- [6] L. Zanotti, “Reti neurali e deep learning: applicazioni commerciali in continua crescita,” 2020, controllata il 28-02-2023. [Online]. Available: <https://www.zerounoweb.it/cio-innovation/>

- reti-neurali-e-deep-learning-applicazioni-commerciali-in-continua-crescita/
(Citato alle pagine iii e 13)
- [7] Wikipedia, "File:artificialneuronmodel english.png — wikipedia, l'enciclopedia libera," 2021, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/File:ArtificialNeuronModel_english.png (Citato alle pagine iii e 14)
- [8] A. Mittal, "Understanding rnn and lstm," 2019, controllata il 28-02-2023. [Online]. Available: <https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e> (Citato alle pagine iii e 22)
- [9] S. Dobilas, "Lstm recurrent neural networks — how to teach a network to remember the past," Available: <https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e>, 2022, [Online], Controllata il 28-02-2023. (Citato alle pagine iii e 23)
- [10] A. Ranjan, "Image captioning with an end-to-end transformer network," 2022, controllata il 28-02-2023. [Online]. Available: <https://python.plainenglish.io/image-captioning-with-an-end-to-end-transformer-network-8f39e1438cd4> (Citato alle pagine iii e 24)
- [11] B. K, "Bert transformers for natural language processing," 2022, controllata il 28-02-2023. [Online]. Available: <https://blog.paperspace.com/bert-natural-language-processing/> (Citato alle pagine iii e 25)
- [12] J. Alammar, "The illustrated gpt-2 (visualizing transformer language models)," 2019, controllata il 28-02-2023. [Online]. Available: <https://jalammar.github.io/illustrated-gpt2/> (Citato alle pagine iii, iv, 26, 27, 28, 29, 30 e 31)
- [13] M. Woolf, "gpt-2-simple," 2021, controllata il 28-02-2023. [Online]. Available: <https://github.com/minimaxir/gpt-2-simple> (Citato alle pagine v, 32, 33 e 37)
- [14] Wikipedia, "Elaborazione del linguaggio naturale — wikipedia, l'enciclopedia libera," 2022, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/Elaborazione_del_linguaggio_naturale (Citato alle pagine 6 e 7)

- [15] T. F. W. Ben Lutkevich, "Definition natural language processing (nlp)," 2023, controllata il 28-02-2023. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP> (Citato a pagina 6)
- [16] A. V. D. RES, "La costruzione di un modello di natural language processing: Dalla raccolta alla pulizia dei dati," Available: <https://res-group.eu/articoli/la-costruzione-di-un-modello-di-natural-language-processing-dalla-raccolta-alla-pulizia-dei-dati>, 2021, [Online], Controllata il 28-02-2023. (Citato a pagina 6)
- [17] ihal, "Che cos'è la comprensione del linguaggio naturale nlu?" 2020, controllata il 28-02-2023. [Online]. Available: <https://ihal.it/che-cose-la-comprensione-del-linguaggio-naturale-nlu/> (Citato alle pagine 7 e 8)
- [18] P. Dotti, "Natural language generation, come andare oltre i chatbot e gli assistenti vocali," 2021, controllata il 28-02-2023. [Online]. Available: <https://www.ai4business.it/intelligenza-artificiale/natural-language-generation-oltre-i-chatbot-e-gli-assistenti-vocali/> (Citato alle pagine 8 e 17)
- [19] A. AWS, "Cos'è il deep learning?" 2023, controllata il 28-02-2023. [Online]. Available: <https://aws.amazon.com/it/what-is/deep-learning/> (Citato alle pagine 9, 10, 11, 13, 16 e 17)
- [20] M. R. Carbone, "Transfer learning, cos'è, come funziona e applicazioni," 2022, controllata il 28-02-2023. [Online]. Available: <https://www.ai4business.it/intelligenza-artificiale/transfer-learning-cose-come-funziona-e-applicazioni/> (Citato a pagina 10)
- [21] F. R. Mashrur, "What is the difference between transfer learning vs fine tuning vs. learning from scratch?" Available: <https://www.researchgate.net/post/What-is-the-difference-between-Transfer-Learning-vs-Fine-Tuning-vs-Learning-from->

- scratch#:~:text=Training%20from%20scratch%20means%20that,a%20lot%20of%20computational%20power., 2020, [Online], Controllata il 28-02-2023. (Citato a pagina 10)
- [22] H. Ampadu, "Dropout in deep learning," 2021, controllata il 28-02-2023. [Online]. Available: <https://ai-pool.com/a/s/dropout-in-deep-learning> (Citato a pagina 10)
- [23] R. Kundu, "Everything you need to know about few-shot learning," 2022, controllata il 28-02-2023. [Online]. Available: <https://blog.paperspace.com/few-shot-learning/> (Citato a pagina 10)
- [24] Wikipedia, "Rete neurale — wikipedia, l'enciclopedia libera," 2021, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/Rete_neurale (Citato alle pagine 12 e 13)
- [25] N. Canu, "Neurone," 2010, controllata il 28-02-2023. [Online]. Available: https://www.treccani.it/enciclopedia/neurone_res-f6daa336-9b53-11e1-9b2f-d5ce3506d72e_%28Dizionario-di-Medicina%29/ (Citato a pagina 12)
- [26] Wikipedia, "Rete neurale artificiale — wikipedia, l'enciclopedia libera," 2023, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/Rete_neurale_artificiale (Citato alle pagine 13, 14 e 15)
- [27] C. Casellato, "Qual è la differenza tra funzione di attivazione e funzione di trasferimento in una rete neurale?" Available: <https://it.quora.com/Qual-%C3%A8-la-differenza-tra-funzione-di-attivazione-e-funzione-di-trasferimento-in-una-rete-neurale>, 2019, [Online], Controllata il 28-02-2023. (Citato a pagina 14)
- [28] NetAi, "Rguida rapida alle funzioni di attivazione nel deep learning," 2021, controllata il 28-02-2023. [Online]. Available: <https://netai.it/guida-rapida-alle-funzioni-di-attivazione-nel-deep-learning/#page-content> (Citato a pagina 15)

- [29] Wikipedia, “Modèle de langage — wikipedia, l’enciclopedia libera,” 2023, controllata il 28-02-2023. [Online]. Available: https://fr.wikipedia.org/wiki/Mod%C3%A8le_de_langage (Citato alle pagine 17 e 20)
- [30] —, “Traduzione automatica — wikipedia, l’enciclopedia libera,” 2022, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/Traduzione_automatica (Citato a pagina 17)
- [31] —, “Spell checker — wikipedia, l’enciclopedia libera,” 2023, controllata il 28-02-2023. [Online]. Available: https://en.wikipedia.org/wiki/Spell_checker (Citato a pagina 17)
- [32] —, “N-gramma — wikipedia, l’enciclopedia libera,” 2020, controllata il 28-02-2023. [Online]. Available: <https://it.wikipedia.org/wiki/N-gramma> (Citato alle pagine 18 e 19)
- [33] E. Santus, “I limiti dei modelli linguistici e la sfida di ai21 labs,” 2022, controllata il 28-02-2023. [Online]. Available: <https://www.notizie.ai/i-limiti-dei-modelli-linguistici-e-la-sfida-di-ai21-labs/> (Citato alle pagine 18 e 19)
- [34] E. Giannini, “Modello linguistico con n-grammi,” 2020, controllata il 28-02-2023. [Online]. Available: <https://enricogiannini.com/15/modello-linguistico-con-n-grammi/> (Citato alle pagine 18 e 19)
- [35] Wikipedia, “Rete neurale feed-forward — wikipedia, l’enciclopedia libera,” 2022, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/Rete_neurale_feed-forward (Citato alle pagine 19 e 20)
- [36] —, “Rete neurale ricorrente — wikipedia, l’enciclopedia libera,” 2022, controllata il 28-02-2023. [Online]. Available: https://it.wikipedia.org/wiki/Rete_neurale_ricorrente (Citato alle pagine 19 e 22)
- [37] —, “Long short-term memory — wikipedia, l’enciclopedia libera,” 2023, controllata il 28-02-2023. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory (Citato alle pagine 20 e 23)

- [38] —, “Transformer (machine learning model) — wikipedia, l’enciclopedia libera,” 2023, controllata il 28-02-2023. [Online]. Available: [https://en.wikipedia.org/wiki/Transformer_\(machine_learning_model\)](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)) (Citato alle pagine 20 e 25)
- [39] P. Sharma, “Feedforward neural network: Its layers, functions, and importance,” 2022, controllata il 28-02-2023. [Online]. Available: <https://www.analyticsvidhya.com/blog/2022/01/feedforward-neural-network-its-layers-functions-and-importance/> (Citato a pagina 20)
- [40] geeksforgeeks, “Introduction to recurrent neural network,” 2022, controllata il 28-02-2023. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/> (Citato a pagina 22)
- [41] S. K. T, “Natural language processing – sentiment analysis using lstm,” 2022, controllata il 28-02-2023. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/06/natural-language-processing-sentiment-analysis-using-lstm/> (Citato a pagina 23)
- [42] Nvidia, “What is a transformer model?” 2022, controllata il 28-02-2023. [Online]. Available: <https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/> (Citato a pagina 25)
- [43] R. Jagtap, “Openai gpt: Generative pre-training for language understanding,” 2020, controllata il 28-02-2023. [Online]. Available: <https://medium.com/dataseries/openai-gpt-generative-pre-training-for-language-understanding-bbbdb42b7ff4> (Citato a pagina 26)
- [44] Wikipedia, “Gpt-3 — wikipedia, l’enciclopedia libera,” 2023, controllata il 28-02-2023. [Online]. Available: <https://it.wikipedia.org/wiki/GPT-3> (Citato a pagina 26)

Ringraziamenti

Quando ho iniziato questo percorso, la prima cosa che pensai, fu: “sarà una passeggiata”. Tendevo sempre a sottovalutare le cose, poiché ero convinto che appena esse si complicavano, iniziavo ad avere il pieno controllo delle mie paure ed ansie, con il solo scopo di riuscire a sfruttare tutte le mie capacità. Posso dire ora, che erano solo le parole di un 19enne meno maturo di oggi (a sua volta meno maturo del suo io del futuro). È stato un percorso molto difficile, che ha richiesto molti sacrifici, portandomi a ripetere la parola “NO” troppe volte, trascinando un ragazzo di 22 anni con un futuro così incerto, verso la seguente domanda: “È stata la scelta giusta?”. Ad essa non so ancora rispondere, ma so di per certo che non ce l’avrei fatta mai, senza chi spendeva una parola per darmi conforto. Al giorno d’oggi credo fortemente che abbiamo bisogno di conforto o di qualcuno che ci faccia ricordare che la felicità la si può trovare anche negli attimi più tenebrosi, se solo accendessimo una luce. Purtroppo, non abbiamo tutti persone del genere al nostro fianco, infatti, c’è gente rimasta in solitudine (pur essendo da un certo punto di vista affascinante), divorata soprattutto dalla pressione sociale, di cui tanto si parla. Per questo ringrazio il professore Fabio Palomba e l’intero personale di Sense Square SRL, che hanno fatto parte di quest’ultima tappa della mia esperienza triennale, dimostrandosi da subito entusiasti e disponibili, con una gentilezza unica piuttosto che rara, da trovare negli ultimi anni. Ringrazio i miei genitori per aver dato una possibilità ai miei obiettivi, significa tanto per me, soprattutto vedere una persona affaticata su una bici, per

permettergli di raggiungere proprio quegli obiettivi. Ringrazio i miei fratelli e Ilaria, che hanno dato attenzione alla mia crescita e a ciò che mi mancava, grazie a voi ho avuto quelle spinte di cui il mio carattere necessita. Ringrazio i miei nonni paterni e le persone che non ci sono più, avete fatto parte del mio passato, un punto di riferimento importante per affrontare le avversità del futuro. Ringrazio i miei nonni materni, tutti i miei zii e cugini, nascendo e crescendo con voi, mi avete formato, dato libertà per il gioco e curato di tutte quelle attenzioni di cui un bambino ha bisogno. Ora tocca ai miei amici, partendo dal primo gruppetto con cui sono nato, Pio, Cicchellino, Luca, Riccardo, ne abbiamo passate tante insieme, affrontando ogni avventura con le nostre bici. Anche se le giornate non sono più come quelle di prima, ci resta un ricordo forte permettendoci di affrontare il nostro futuro. Subito dopo, si sono ricostruite a causa dell'età, delle amicizie d'infanzia come Gennaro e Luigi, li ringrazio per aver portato rispettivamente razionalità ed il miglior umorismo italiano. Col passare del tempo, si è costruito il gruppo che ora tutti voi conoscete con Bruna, Federica, Adriana, Eduard, Clara, Annamaria, Aurora, Lisa, Martina, a tal punto da sopportarci anche per condividere una casa per le vacanze fatte. Un fortissimo grazie, va al mio amico Gibaldo, per aver colto una vera amicizia, nata in un contesto scolastico, ma chi è che riesce a coltivare delle amicizie del genere in determinati contesti? Ringrazio Simone, Rocco, Antonio di nola e Antonio di solofra (giusto per differenziarli), componenti di raaf_gaming, ci siamo fatti forza a vicenda, studiando per 3 anni insieme, i miei risultati pratici sono solo una parte dei progetti affrontati insieme. Ringrazio inoltre la mia piccola cagnolina Trilli, perché dopo una lunga sessione di studio, stritolargli le orecchie, eliminava in me tutto lo stress accumulato. Infine spero di essere riuscito a trasmettervi più una riflessione verso tutte quelle persone che non ce l'hanno fatta, definendosi dei fallimenti, invece di raccontarvi delle storie passate insieme, quando già con il ricordo lo facciamo.