UNIVERSITÀ DEGLI STUDI DI UDINE
HIC SUNT FUTURA

DIPARTIMENTO DI SCIENZE MATEMATICHE, INFORMATICHE E FISICHE

TESI DI LAUREA IN
INFORMATICA

Codifica di Referti Medici Italiani con NLP: Un Confronto tra BERT e Baseline per la Classificazione SNOMED

CANDIDATO
Enrico Tazzer

RELATORE
Prof. Vincenzo Della Mea

CORRELATORE
Prof. Kevin Roitero

Anno accademico 2023-2024

Contatti dell'Istituto
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

Sommario

Nel corso dell'ultimo decennio, lo sviluppo di algoritmi e modelli di apprendimento automatico ha profondamente trasformato la nostra quotidianità. Un esempio emblematico di questo cambiamento è rappresentato dall'introduzione di **ChatGPT**, che ha avuto un impatto significativo sul nostro modo di vivere, contribuendo, in alcuni casi, a migliorarlo. Quando abbiamo dei dubbi e ne cerchiamo la risposta, apriamo ChatGPT e poniamo direttamente il nostro quesito all'agente, come se ci rivolgessimo a una persona fisicamente presente. Qualche anno fa, invece, facevamo affidamento su Google per trovare risposte, navigando tra vari siti web fino a reperire la nostra risposta.

Anche l'industria sta evolvendo per sfruttare queste nuove risorse, e non è insolito vedere dispositivi elettronici promossi come "intelligenti", seguendo la tendenza dell'uso del termine **AI** (Artificial Intelligence). Questa strategia si rivela efficace nel catturare l'interesse dei consumatori in modo astuto.

Altri settori stanno incontrando maggiori difficoltà nell'affrontare questa "evoluzione digitale". Questa tesi si propone di analizzarne uno in particolare: il **settore sanitario**, in cui la digitalizzazione della documentazione clinica rappresenta una sfida particolarmente complessa.

Il principale ostacolo alla digitalizzazione del sistema sanitario risiede nell'assenza di standard nazionali condivisi per la creazione e gestione dei documenti clinici, nonché di un software comune a livello nazionale. Attualmente, gli standard variano da regione a regione e, in alcuni casi, persino da una struttura sanitaria all'altra, limitando significativamente l'interoperabilità tra i diversi sistemi. Tale frammentazione complica sia l'archiviazione sia l'accesso alla documentazione dei pazienti, ostacolando l'efficienza complessiva del sistema.

In questo contesto una possibile soluzione potrebbe essere portata dall'implementazione del **Fascicolo Sanitario Elettronico** (FSE)[27], operazione che al giorno d'oggi è un miraggio. L'FSE offre sulla carta numerosi servizi vantaggiosi (prenotazioni di visite e analisi, archivio unico della documentazione sanitaria, consultazione di referti, ecc...), che porterebbero ad un incremento dell'efficienza del servizio sanitario italiano, che non brilla sotto questo aspetto. Questo strumento sarebbe un punto di svolta verso la medicina preventiva[32], disciplina che grazie allo sviluppo di sistemi di intelligenza artificiale sta ottenendo ottimi risultati, se non fosse per il suo scarso utilizzo da parte delle strutture[24].

L'obiettivo di questa tesi, in collaborazione con l'Anatomia Patologica dell'IRCCS San Gerardo di Monza, sede del Corso di Laurea in Medicina dell'Università Milano-Bicocca, è quello di studiare un problema in ambito sanitario, nello specifico la codifica non uniforme dei referti, e fornire uno strumento risolutivo all'ostacolo studiato. Tratteremo quindi di modelli e algoritmi di **machine e deep learning** al fine di **codificare referti** utilizzando una codifica standardizzata, lo **SNOMED-3**.

Indice

Elenco delle tabelle

Elenco delle figure

1

Natural Language Processing (NLP)

Il **Natural Language Processing** è una branca del machine learning che mira a rendere i calcolatori in grado di comprendere e utilizzare il linguaggio umano per interagire, aprendo così la strada all'era dell'intelligenza artificiale generativa.

Le applicazioni pratiche di questa disciplina ricoprono numerose sfere della nostra quotidianità. Una delle applicazioni più comuni è nei motori di ricerca, dove il NLP viene utilizzato per raffinare la comprensione della richiesta dell'utente, fornendo inoltre risultati più precisi. Strumenti come Google Translate sfruttano modelli e architetture avanzate per rendere i loro servizi più accurati ed efficaci. Il Natural Language Processing è fondamentale negli assistenti virtuali e chatbot come Alexa e Siri, il cui compito è quello di comprendere ed elaborare i comandi vocali impartiti dall'utente per poi rispondere utilizzando una voce virtuale, e quindi generando del testo da riprodurre.

I modelli di NLP, tuttavia, sono ancora lontani dalla perfezione, in gran parte a causa della natura intrinsecamente error prone del linguaggio umano. Questo comporta una serie di sfide che la ricerca nel settore sta cercando di superare. Tra queste figurano la continua evoluzione della lingua, con l'introduzione di nuovi termini, e il ruolo del tono vocale, che può alterare significativamente il significato semantico delle parole.

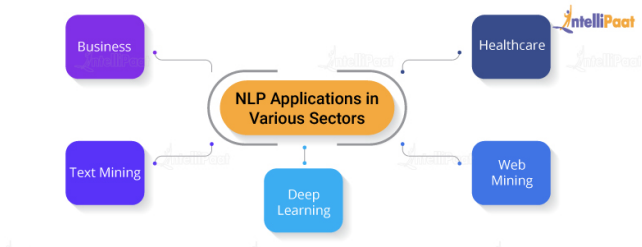

Figura 1.1: Principali applicazioni di NLP (da [19])

1.1 Approcci per un sistema NLP

NLP combina la potenza della linguistica computazionale[1] con sistemi di machine learning, per ottenere una rappresentazione del testo, o del parlato, comprensibile da uomo e macchina. In questo modo il computer è in grado di utilizzare questa rappresentazione per la risoluzione del task, in tal senso vi sono tre approcci principali per la creazione di un sistema NLP:

- **Rule-Based NLP**: primo approccio utilizzato per l'analisi del linguaggio naturale che si basava su semplici **alberi decisionali**, progettati per rispondere a prompt specifici attraverso regole predefinite. Sebbene fosse efficace per le circostanze in cui veniva progettato, questo metodo non consentiva l'uso di algoritmi di machine learning, risultando quindi estremamente rigido e non scalabile. La mancanza di adattabilità ai cambiamenti del linguaggio e l'incapacità di apprendere dai dati rendevano questo approccio inefficiente per applicazioni più complesse e dinamiche

- **Statistical NLP**: si basa sull'estrazione, classificazione ed etichettatura automatica degli elementi del linguaggio dai dati di input, assegnando a ciascun possibile significato una probabilità statistica. Questo approccio ha introdotto la codifica degli elementi del linguaggio in **rappresentazioni vettoriali**, consentendo di modellare il linguaggio utilizzando metodi matematici e statistici, come la regressione o i modelli di **Markov**[2]. Grazie a questo, è possibile applicare algoritmi di machine learning per l'analisi e la previsione, come **Support Vector Machines** (SVM), la **Logistic Regression** e **Random Forest**, che migliorano la capacità dei modelli NLP di gestire dati complessi e dinamici.

- **Deep Learning NLP**: rappresenta un'estensione dell'approccio statistico, sfruttando **reti neurali** per migliorare la comprensione e la generazione del linguaggio naturale. A differenza degli approcci sopracitati, che si basano su regole e probabilità statistiche, i modelli di deep learning apprendono rappresentazioni più complesse del linguaggio attraverso **grandi quantità di dati**, catturando contesti e relazioni tra parole con maggiore precisione. Nei capitoli successivi esploreremo alcuni di questi modelli, come **BERT** (Bidirectional Encoder Representations from Transformers), **RNN** (Recurrent Neural Networks) e **GRU** (Gated Recurrent Units), che hanno rivoluzionato il campo del NLP, migliorando le prestazioni in compiti come il riconoscimento di entità, la traduzione automatica e la generazione di testi.

1.2 Tecniche di NLP

In questa sezione verranno approfondite le principali tecniche di elaborazione del linguaggio naturale (NLP), impiegate per analizzare, comprendere e generare automaticamente il linguaggio umano. Nel corso degli anni, tali tecniche hanno subito un'evoluzione significativa grazie ai progressi del machine learning, consentendo lo sviluppo di modelli sempre più accurati e complessi. Questi avanzamenti hanno migliorato la capacità delle macchine di gestire dati linguistici, rendendo possibili applicazioni più sofisticate. In chiusura del capitolo, verranno citati alcuni dei modelli più avanzati, che rappresentano lo stato dell'arte nel campo del NLP[12].

[1] Disciplina che si concentra sullo sviluppo di formalismi descrittivi del linguaggio naturale, affinché possano essere tradotti in programmi eseguibili dal calcolatore.

1.2.1 Tokenization

La tecnica più semplice, ancora ampiamente utilizzata, è la **tokenizzazione**. Questo processo consiste nella suddivisione di un testo in unità più piccole (token). Ci sono tre tipologie di tokenization, che si differenziano per la dimensione dei token generati, e sono tokenization a livello di **parola**, **sotto-parola** e **carattere**. L'utilizzo di ciascuna tipologia dipende dal contesto applicativo, ad esempio la tokenizzazione a livello di carattere è frequentemente impiegata quando è necessario trattare lingue asiatiche.

1.2.2 Lemmatization e Stemming

Le tecniche di **lemmatizzazione** e **stemming** vengono utilizzate per ridurre le parole alla loro **radice o forma base**. L'utilizzo di queste tecniche è fondamentale nell'apprendimento del linguaggio poiché riducono la variabilità lessicale, portando le varie forme di una parola, come ad esempio "portone" e "porticina", ad una singola forma base, in questo esempio quindi "porta", semplificando l'analisi complessiva del linguaggio. Stemming e lemmatization differiscono nel modo in cui riducono la parola (vedi figura 1.2). Il primo tronca semplicemente le ultime lettere, perdendo in molti casi il significato della parola, ad esempio "correre" viene ridotto a "corr". Questa approccio è più semplice e computazionalmente leggero, per cui adatta a dataset voluminosi. La lemmatization, invece, riduce la parola al suo lemma, ovvero la parola che si trova nel dizionario, ottenendo un risultato dal punto di vista semantico più corretto, ma computazionalmente oneroso. L'utilizzo di una delle due tecniche rispetto all'altra non porta ad un aumento significativo delle prestazioni, per la scelta è prettamente soggettiva e contestuale all'uso.

1.2.3 Bag of Words (BoW)

Il modello **Bag of Words** (BoW) è una tecnica di rappresentazione testuale comunemente utilizzata per la classificazione di documenti. Questo modello trasforma un testo in un insieme (o "borsa") di parole, ignorando l'ordine in cui appaiono e trattandole come elementi indipendenti. Ogni parola nel documento viene associata a un indice unico, formando un **vocabolario**. Successivamente, per rappresentare il documento, viene costruito un vettore di dimensione pari al vocabolario, in cui ogni posizione corrisponde a una parola dello stesso, e il valore nella posizione rappresenta la **frequenza** con cui quella parola appare nel testo.

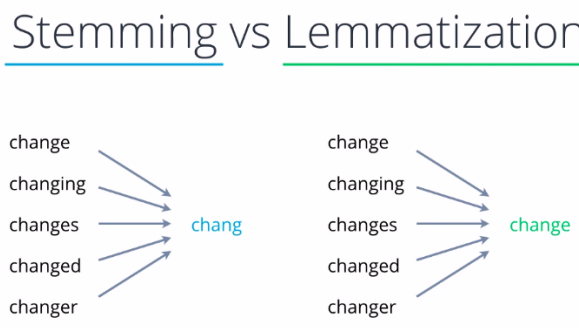

Figura 1.2: Lemmatization vs. Stemming (da [5])

Tuttavia, nonostante la sua semplicità ed efficacia in molti contesti, il modello BoW ha un notevole svantaggio: **perde completamente l'informazione semantica**. Poiché ignora l'ordine e il contesto in cui le parole appaiono, il modello non tiene conto delle relazioni semantiche tra le parole, come la sintassi o il significato che emerge dall'interazione tra termini nel testo. Di conseguenza, testi che utilizzano le stesse parole ma in ordini differenti, o con sfumature contestuali diverse, vengono trattati come identici, portando a una riduzione della capacità del modello di catturare il significato profondo del testo.

1.2.4 TFIDF

Il **TF-IDF** (Term Frequency-Inverse Document Frequency) è una tecnica avanzata di pesatura delle parole utilizzata per valutare l'importanza di una parola in un documento all'interno di un **corpus** (racolta di più documenti). A differenza del modello Bag of Words, che conta semplicemente la frequenza delle parole, il TF-IDF considera non solo quante volte una parola appare in un singolo documento (Term Frequency, TF), ma anche quante volte quella stessa parola compare in tutto il corpus (Inverse Document Frequency, IDF). La formula di TF-IDF è la seguente:

$$\text{TF-IDF}(w, d) = tf_{w,d} \times \log\left(\frac{N}{df_w}\right)$$

dove:

- **N**: numero di documenti del corpus
- $\mathbf{df_w}$: numero di documenti in cui appare la parola w
- $\mathbf{tf_{w,d}}$: La frequenza della parola w nel documento d.

Questo approccio permette di dare più peso alle parole che sono frequenti in un singolo documento, ma non comuni nel resto del corpus, rendendole così più rappresentative del contenuto del documento. Grazie a questa caratteristica, il TF-IDF è ampiamente utilizzato in applicazioni come il text mining, la ricerca di informazioni e i motori di ricerca, in cui è necessario classificare o confrontare documenti basati sul loro contenuto testuale

1.2.5 Word Emebedding

Il **word embedding** è una tecnica utilizzata nel Natural Language Processing (NLP) per rappresentare le parole in uno **spazio vettoriale continuo**, in modo che le parole con significati simili siano rappresentate da vettori vicini tra loro. A differenza del modello Bag of Words, che rappresenta le parole come indici o vettori di frequenze, i word embedding catturano le **relazioni semantiche** tra le parole, considerando il contesto in cui esse appaiono.

Uno degli approcci più noti per creare word embedding è il modello **Word2Vec**, che utilizza reti neurali per addestrare i vettori di parole in modo tale che parole con un contesto simile abbiano vettori simili. Altri modelli di embedding popolari includono **GloVe** (Global Vectors for Word Representation) e i modelli basati su trasformatori come BERT.

I word embedding offrono vantaggi significativi, poiché consentono di catturare non solo la somiglianza tra le parole, ma anche relazioni più complesse, come quelle tra sinonimi, contrari o persino

associazioni semantiche più profonde. Questo ha reso gli embedding particolarmente utili in molte applicazioni, come la traduzione automatica, l'analisi del sentiment e la ricerca semantica.

1.2.6 Modelli preaddestrati (BERT, GPT, T5)

I **modelli preaddestrati**[29] sono modelli di **intelligenza artificiale** che sono stati addestrati su grandi quantità di dati testuali prima di essere utilizzati per compiti specifici di Natural Language Processing, come la classificazione dei testi, la traduzione automatica, o la generazione di linguaggio naturale. Questi modelli vengono preaddestrati su collezioni di testi generici e poi affinati per compiti più specifici tramite un processo di fine-tuning su set di dati mirati.

Uno degli approcci più avanzati in questo ambito è quello dei modelli basati su trasformers[34], come **BERT** (Bidirectional Encoder Representations from Transformers), **GPT**[36] (Generative Pre-trained Transformer), e **T5**[30] (Text-To-Text Transfer Transformer). Questi modelli sfruttano l'architettura a **trasformatori** per gestire in modo efficace la comprensione del contesto e delle relazioni tra parole, migliorando notevolmente le prestazioni rispetto ai modelli precedenti.

I modelli preaddestrati offrono diversi vantaggi:

- **Riduzione del tempo di addestramento**: Poiché sono già stati pre-addestrati su enormi quantità di dati, richiedono meno risorse computazionali per essere adattati a compiti specifici.
- **Capacità di generalizzazione**: Questi modelli possono essere adattati a una vasta gamma di applicazioni NLP, come il riassunto automatico, la traduzione e la risposta a domande.
- **Comprensione contestuale avanzata**: Grazie all'addestramento su grandi quantità di dati, i modelli preaddestrati riescono a catturare il contesto in modo più efficace rispetto alle tecniche tradizionali.

La loro applicazione ha rivoluzionato il campo dell'NLP, permettendo di ottenere prestazioni allo **stato dell'arte** su molti compiti linguistici complessi

2

Analisi approfondita e definizione del problema

Dopo aver esplorato il concetto di Natural Language Processing e aver compreso i suoi aspetti fondamentali, possiamo ora concentrarci sul suo utilizzo pratico, analizzando come queste tecnologie possano essere applicate in un **contesto reale** per risolvere un compito specifico.

2.1 Obiettivo

Il progetto si proponeva di automatizzare la codifica di codici **SNOMED-3** ai referti medici utilizzando tecniche avanzate di Natural Language Processing (NLP) e **Large Language Models** (LLMs). In particolare, il compito consisteva nell'addestrare e testare modelli NLP con un dataset di circa 400mila referti medici, ciascuno già etichettato con un codice SNOMED-3.

L'obiettivo principale era misurare la capacità del sistema di **codificare automaticamente i referti medici** ai corretti codici SNOMED-3, migliorando l'efficienza e l'accuratezza del processo rispetto ai metodi manuali tradizionali. Questo approccio potrebbe ridurre significativamente il carico di lavoro umano, velocizzando la classificazione dei referti medici e garantendo una maggiore coerenza e precisione nell'associazione dei codici SNOMED-3

2.2 Perché affrontare la tematica

La **digitalizzazione della sanità** ha portato enormi potenziali benefici, come una maggiore efficienza operativa, una migliore accessibilità ai dati clinici e un'assistenza sanitaria più personalizzata. Tuttavia, ha anche evidenziato diverse problematiche che il settore deve affrontare. Uno dei principali problemi riguarda la **frammentazione dei sistemi informativi sanitari**, con piattaforme e software che spesso non sono integrati tra loro, ostacolando lo scambio fluido di dati tra ospedali, cliniche e altri operatori sanitari. Questo può rallentare il processo decisionale e ridurre la qualità delle cure.

Un altro aspetto critico è la questione della sicurezza e della **privacy dei dati**. Con la digitalizzazione, enormi quantità di informazioni sensibili sui pazienti vengono archiviate e condivise elettronicamente, aumentando il rischio di violazioni della sicurezza informatica e del furto di dati. Inoltre, l'adozione di

nuove tecnologie richiede competenze digitali che molti operatori sanitari potrebbero non avere, causando resistenza al cambiamento e problemi nell'implementazione di sistemi innovativi. Infine, la **mancanza di standardizzazione** nei formati e nei protocolli utilizzati per la digitalizzazione dei dati sanitari rende difficile creare un'infrastruttura sanitaria digitale coesa e universalmente accessibile.

3

Modelli

In questo capitolo esamineremo i diversi modelli impiegati per affrontare il task di **classificazione**, analizzando sia gli approcci tradizionali di machine learning sia le architetture più avanzate basate su reti neurali. Ogni modello verrà descritto in termini di funzionamento, vantaggi e limiti, con particolare attenzione alle tecniche di Natural Language Processing che si dimostrano fondamentali per la classificazione di dati testuali.

3.1 Baselines

La **baseline** nel contesto del machine learning e del NLP rappresenta un modello di riferimento o un metodo di base utilizzato per confrontare e valutare le prestazioni di modelli più avanzati.

In questo caso, sono stati impiegati modelli e algoritmi standard di NLP che riflettono lo stato dell'arte. Si tratta di sistemi relativamente semplici, la cui efficacia è stata ampiamente dimostrata nel corso degli anni. Lo scopo principale di questi modelli baseline è fornire un quadro di riferimento per comprendere meglio la complessità del task, senza necessariamente raggiungere le prestazioni ottimali, ma piuttosto offrendo un confronto utile per valutare eventuali miglioramenti apportati da tecniche più sofisticate.

3.1.1 Logistic Regression

La **logistic regression**[11] è un algoritmo di supervised learning utilizzato per problemi di classificazione, specialmente nella classificazione binaria. A differenza della regressione lineare, che prevede come output un valore continuo, la logistic regression stima la probabilità che un dato appartenga a una delle due classi utilizzando la funzione **sigmoide**.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Questa funzione comprime il valore calcolato in un intervallo compreso tra 0 e 1, interpretato come la probabilità che l'input appartenga alla classe positiva. In base a una soglia (solitamente 0,5), il modello assegna l'input a una delle due classi: se la probabilità è superiore o uguale a 0,5, il risultato sarà la classe 1; altrimenti, la classe 0.

Inoltre, può essere estesa per problemi di **classificazione multi-classe** attraverso l'uso della funzione **softmax**, che permette di gestire più di due classi:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K,$$

dove z_i indica la probabilità di appartenenza alla i-esima classe.

La logistic regression è apprezzata per la sua semplicità ed efficienza, anche su dataset di grandi dimensioni, e permette una facile interpretazione dei risultati. Tuttavia, non è adatta a problemi complessi o non lineari e può essere sensibile agli outlier. Trova ampia applicazione in settori come la **diagnosi medica** (previsione di malattie basata su dati clinici)[18], la previsione del rischio finanziario e il marketing.

3.1.2 Random Forest

La **Random Forest**[6] è un algoritmo di supervised learning basato su un insieme di **alberi decisionali**, utilizzato per risolvere problemi sia di classificazione che di regressione. L'idea centrale dietro Random Forest è quella di combinare più alberi decisionali indipendenti, creando così una "foresta" di alberi, dove ogni albero viene addestrato su un diverso sottoinsieme del dataset originale, selezionato casualmente tramite un processo chiamato **bootstrap sampling**. Oltre alla casualità nel campionamento dei dati, viene introdotta ulteriore variabilità nella scelta delle feature per ogni albero. Questo doppio livello di casualità contribuisce a rendere il modello più robusto e preciso.

L'output finale della Random Forest viene determinato tramite votazione a maggioranza nel caso della classificazione (viene scelta la classe più votata dagli alberi) o tramite la media delle previsioni per i problemi di regressione.

Grazie a queste caratteristiche, la Random Forest è particolarmente **robusta all'overfitting** rispetto ai singoli alberi decisionali. Tuttavia, uno dei suoi limiti principali è l'elevato **costo computazionale**: il numero elevato di alberi richiesto per migliorare la precisione del modello può rallentare sia il processo di addestramento che quello di inferenza, rendendola meno adatta per grandi dataset.

Nonostante ciò, la Random Forest è un potente strumento e trova applicazioni pratiche in vari settori, spesso come baseline, ad esempio nella diagnosi medica[1], nel riconoscimento facciale, e nel rilevamento di frodi.

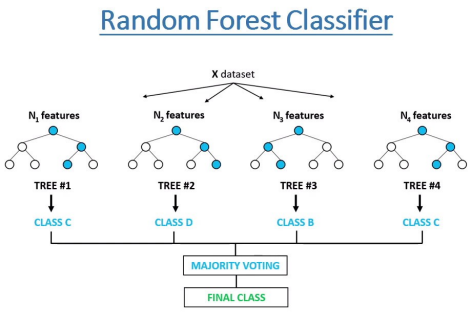

Figura 3.1: Random Forest (da [20])

3.1.3 Support Vector Machines (SVMs)

La **Support Vector Machine**[10] è un algoritmo di supervised learning utilizzato per la classificazione dei dati. Il suo funzionamento si basa sulla ricerca di una **retta** o di un **iper-piano** ideale che massimizzi la distanza tra le diverse classi in uno spazio a dimensionalità N. L'obiettivo è trovare il **margine massimo** tra i dati di classe diversa, rendendo il modello robusto e meno incline a errori.

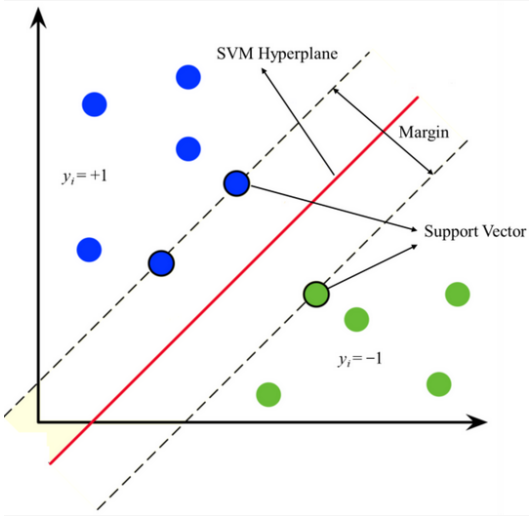

Figura 3.2: Support Vector Machine (da [35])

Le SVMs sono comunemente impiegate per problemi di classificazione, in particolare per la classificazione binaria, sia in contesti lineari che non lineari. Funzionano sotto l'assunzione che i dati siano **linearmente separabili**. Tuttavia, quando questa condizione non è soddisfatta, viene utilizzata una tecnica chiamata **kernel trick**. Questo approccio si basa su funzioni kernel che trasformano i dati in uno spazio a dimensionalità superiore, dove è possibile trovare una separazione lineare. Tra i kernel più usati ci sono il kernel polinomiale e il kernel radiale (RBF).

Nel nostro caso, trattandosi di un problema di classificazione multi-classe, sono richieste estensioni delle SVM[17]. Tra gli approcci più comuni ci sono:

- **One-vs-Rest** (OvR): addestra un classificatore per ogni classe, confrontandola con tutte le altre.
- **One-vs-One** (OvO): addestra un classificatore per ogni coppia di classi.
- **Directed Acyclic Graph** (DAG): utilizza una struttura ad albero per combinare i risultati dei classificatori binari.

Questi approcci permettono di adattare le SVM anche ai compiti di classificazione con più classi, migliorando l'efficacia in contesti complessi.

Come abbiamo visto la SVM risulta particolarmente adatta a problemi di classificazione grazie alle sue caratteristiche. Tra i principali, spicca l'efficienza nella separazione delle classi, poiché le SVM cercano di massimizzare il margine tra le classi, riducendo così il rischio di errore. Inoltre, le SVM sono efficaci in **spazi ad alta dimensionalità**, come nel Natural Language Processing (NLP), e il kernel trick consente di gestire problemi non lineari codificando i dati in uno spazio di dimensioni superiori.

Tuttavia, le SVM presentano anche alcuni limiti significativi. La **scalabilità** tra i più significativi, richiedendo notevoli risorse in termini computazionali e di tempo su grandi quantità di dati. Inoltre,

le SVM possono risultare difficili da interpretare, soprattutto quando si utilizzano kernel non lineari. Anche la loro sensibilità alla scelta del kernel e dei relativi parametri può rappresentare una sfida, poiché trovare la configurazione ottimale spesso richiede un tuning accurato. Infine, nei problemi di classificazione multi-class, l'utilizzo di metodi come One-vs-One o One-vs-Rest può risultare complesso ed inefficace su dataset molto ampi.

Nel nostro campo di studio, le SVM sono impiegate in vari compiti grazie alla loro capacità di gestire dati ad alta dimensionalità, come i testi rappresentati tramite Bag of Words o TF-IDF. Alcuni esempi di applicazione includono la **classificazione di testi** (spam detection, sentiment analysis), il **Named Entity Recognition** (NER), il rilevamento del linguaggio e il riconoscimento del parlato. Sebbene le SVM siano state una scelta popolare in passato, modelli più recenti come le reti neurali stanno progressivamente sostituendole, specialmente per task più complessi e su larga scala.

3.1.4 Convolutional Neural Network (CNN)

Le **Convolutional Neural Networks** (CNN)[26] sono una classe di **reti neurali** progettate per processare dati con una struttura a griglia, come le immagini. Le CNN sono ampiamente utilizzate in **computer vision** per la loro capacità di estrarre automaticamente pattern rilevanti dai dati visivi, senza richiedere intervento manuale nella selezione delle caratteristiche.

Il funzionamento delle CNN si ispira al modo in cui il cervello umano elabora le immagini. Ogni **neurone** nella CNN è connesso a una specifica area limitata del dato di input, chiamata campo recettivo, e risponde agli stimoli provenienti solo da quella regione. Questi neuroni sono organizzati in strati, e man mano che si procede attraverso gli strati della rete, i neuroni sono in grado di riconoscere pattern via via più complessi. Nei primi strati, la rete rileva caratteristiche semplici come linee, bordi e angoli, mentre negli strati successivi viene eseguita la composizione di pattern più elaborati, fino a riconoscere oggetti complessi come volti o elementi distintivi di un'immagine. Questo processo gerarchico permette alle CNN di analizzare e interpretare le immagini in modo efficace, riducendo la necessità di pre-elaborazione dei dati e migliorando le prestazioni in compiti come la classificazione di immagini, il riconoscimento facciale e la **segmentazione delle immagini**[31].

Tipicamente, una CNN è composta da tre tipi principali di layer: convolutional, pooling e fully connected (vedi figura 3.3).

- **Convolutional layer**: Questo è il blocco centrale della CNN, responsabile dell'operazione di **convoluzione** sui dati di input (solitamente un'immagine). La convoluzione consiste nel calcolo del prodotto scalare tra una porzione dell'immagine e una matrice di parametri addestrabili, chiamata **kernel** (o filtro). Il filtro scorre lungo l'immagine, calcolando il prodotto matriciale

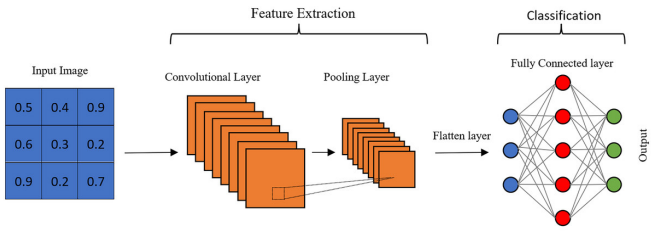

Figura 3.3: Convolutional Neural Network (da [3])

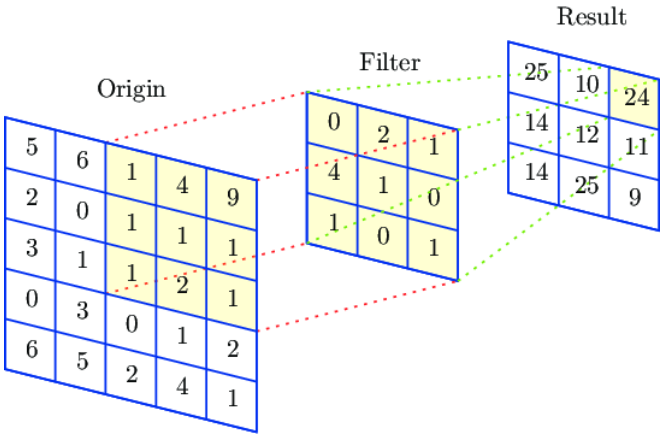

Figura 3.4: Processo di convoluzione (da [15])

in diverse posizioni. Il risultato di questa operazione è una **feature map**, una rappresentazione bidimensionale dell'immagine che evidenzia determinate caratteristiche (o feature) rilevanti. Più filtri vengono applicati in parallelo, ciascuno destinato a rilevare specifici pattern, come bordi, angoli o texture.

- **Pooling layer**: Dopo il convolutional layer, il pooling layer riduce la dimensionalità della feature map, mantenendo le informazioni più rilevanti. L'operazione più comune è il **max pooling**, in cui un filtro scorre sulla feature map e seleziona il valore massimo all'interno di una porzione di dati. Questo processo riduce la dimensione dei dati e consente di diminuire il numero di parametri, rendendo il modello più efficiente e riducendo i tempi di addestramento, senza perdere le informazioni chiave.

Questo processo di estrazione progressiva delle feature, detto **feature learning**, può essere ripetuto più volte per catturare livelli di dettaglio sempre più complessi a seconda della complessità del task e della profondità della rete.

- F**ully connected layer**: Dopo i livelli di convoluzione e pooling, i dati devono essere **appiattiti** (convertiti da una matrice 2D a un vettore 1D) per essere passati al fully connected layer. Qui ogni neurone è connesso a tutti i neuroni dello strato successivo, come nelle reti neurali tradizionali. Il fully connected layer elabora le feature estratte e ne combina le informazioni per ottenere l'output finale. Se l'obiettivo del modello è la classificazione delle immagini, l'output sarà un vettore di probabilità, che rappresenta la probabilità che l'immagine appartenga a ciascuna delle classi. Se invece il task è la segmentazione delle immagini, l'output sarà una segmentation map, in cui ogni pixel viene associato a una classe specifica.

Questa architettura offre numerosi vantaggi nel campo della computer vision, grazie alla sua capacità di apprendere autonomamente le caratteristiche rilevanti delle immagini. Un aspetto chiave è la sua **invarianza alla traslazione**, che consente alle CNN di riconoscere oggetti indipendentemente dalla loro posizione all'interno di un'immagine, rendendo il modello molto efficace. Tuttavia, nonostante le loro eccellenti prestazioni, queste reti presentano alcune limitazioni significative. Tra le principali vi è la necessità di grandi quantità di dati per l'addestramento, spesso supportata da tecniche come la **data augmentation**. Inoltre, architetture complesse tendono a soffrire di **overfitting**, a causa dell'elevato

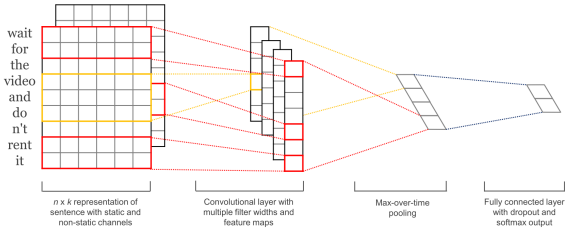

Figura 3.5: CNN in contesto di NLP (da [7])

numero di parametri addestrabili. Un'altra criticità risiede nella limitata interpretabilità del modello, spesso considerato una "**black box**" poiché risulta difficile comprendere appieno il processo decisionale interno.

Le applicazioni delle CNN sono estremamente versatili: oltre alla computer vision, trovano ampio impiego nel settore medico, dove la segmentazione e l'analisi delle immagini possono rilevare formazioni tumorali e migliorare la precisione dei trattamenti medici. Un altro settore cruciale è quello della guida autonoma, dove aziende come Tesla utilizzano queste reti per il riconoscimento di ostacoli e segnali stradali, oltre che per la segmentazione del manto stradale in corsie, necessaria per i sistemi di **lane assistance**[4].

Le CNN si estendono inoltre ad altri campi, tra cui l'elaborazione del linguaggio naturale (NLP)[21]. L'utilizzo della convoluzione, e di conseguenza delle CNN per la classificazione di testi, si basa sull'impiego del **word embedding**, come descritto nel Capitolo 1. Il word embedding fornisce una rappresentazione matriciale del testo, in cui ogni riga corrisponde alla rappresentazione semantica di una parola. Applicando l'operazione di convoluzione, come visto in precedenza, è possibile identificare pattern locali tra parole o gruppi di parole. Successivamente, attraverso il pooling, vengono estratte le caratteristiche più rilevanti dal testo, che vengono poi trasmesse allo strato fully-connected, dove avviene la classificazione finale (vedi figura 3.5).

3.1.5 Recurrent Neural Network (RNN)

La **Recurrent Neural Network** (RNN)[33] è una tipologia di rete neurale progettata per elaborare **dati sequenziali** o **serie temporali**. A differenza delle reti neurali tradizionali, in cui gli input e gli output sono trattati come indipendenti, nelle RNN l'output prodotto dipende non solo dall'input corrente, ma anche dagli elementi precedenti della sequenza. Nelle Bidirectional RNN, l'output può inoltre dipendere dagli elementi futuri, rendendo l'ordine e il contesto degli input fondamentali per la comprensione della sequenza. Questo approccio consente alle RNN di catturare relazioni temporali e dipendenze nel tempo, rendendole particolarmente adatte per compiti come l'elaborazione del linguaggio naturale, la traduzione automatica e il riconoscimento vocale.

La caratteristica fondamentale di una Recurrent Neural Network (RNN) è la presenza di un ciclo nell'architettura, che consente alla rete di mantenere una **memoria a breve termine**. In pratica, l'output dell'iterazione precedente (h_{t-1}) viene utilizzato in combinazione con l'input dell'iterazione corrente (x_t), permettendo alla rete di "ricordare" informazioni rilevanti dai passaggi precedenti della sequenza.

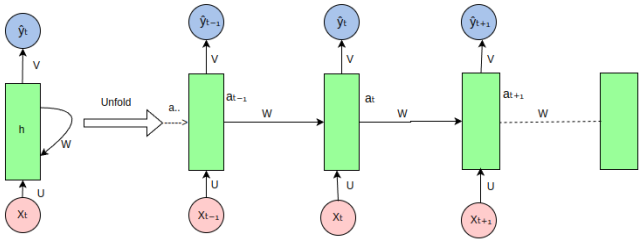

Figura 3.6: Architettura RNN (da [28])

Questo meccanismo di retroazione è ciò che rende le RNN adatte a catturare dipendenze temporali o sequenziali nei dati, come testi, serie temporali o segnali audio.

Come nelle reti neurali classiche, anche le RNN vengono addestrate tramite backpropagation con l'obiettivo di minimizzare la funzione di loss. Tuttavia, per le RNN si utilizza una variante chiamata **Backpropagation Through Time** (BPTT). In questa tecnica, i gradienti vengono propagati all'indietro attraverso l'intera sequenza temporale, aggiornando i pesi in modo da ridurre l'errore complessivo per l'intera sequenza. Questo approccio consente di considerare non solo l'errore del singolo step, ma anche l'influenza che gli step precedenti hanno avuto sull'output finale, rendendo il modello in grado di apprendere dipendenze temporali complesse.

Il principale problema delle RNN "vanilla" è il **vanishing gradient problem** (scomparsa del gradiente). Durante il processo di Backpropagation Through Time (BPTT), i gradienti che vengono propagati all'indietro possono diventare molto grandi o molto piccoli, portando a instabilità nell'apprendimento. In particolare, quando i gradienti diventano troppo piccoli, la rete fatica ad apprendere le dipendenze a lungo termine, cioè ha difficoltà nel mantenere in memoria eventi passati che potrebbero essere cruciali per predizioni future. Questo limita la capacità delle RNN di gestire informazioni di lunga durata all'interno di una sequenza.

Nonostante queste limitazioni, le Recurrent Neural Networks sono ampiamente utilizzate per la loro abilità nel trattare dati sequenziali e serie temporali. Hanno applicazioni in diversi settori, tra cui l'elaborazione del linguaggio naturale (NLP), dove, grazie a varianti più avanzate, giocano un ruolo centrale nello stato dell'arte. Esempi noti del loro utilizzo includono sistemi come Google Translate e Siri, dove le RNN sono fondamentali per la comprensione e la traduzione del linguaggio.

3.1.6 Long Short-Term Memory (LSTM)

La **Long Short-Term Memory** (LSTM)[16] è un'architettura di RNN sviluppata da Sepp Hochreiter e Jürgen Schmidhuber nel 1997. Questa architettura è progettata per superare i limiti delle RNN classiche, in particolare il problema della **scomparsa del gradiente**. Le LSTM si basano su un'**unità di memoria** chiamata unità LSTM, che è composta da quattro Feedforward Neural Networks (FNN) completamente connesse. Tra queste, tre sono responsabili della gestione delle informazioni e vengono definite rispettivamente: forget gate, input gate e output gate.

- Il **forget gate** si occupa della cancellazione delle informazioni non più rilevanti dalla memoria.
- L'**input gate** gestisce l'inserimento di nuove informazioni rilevanti nella memoria.

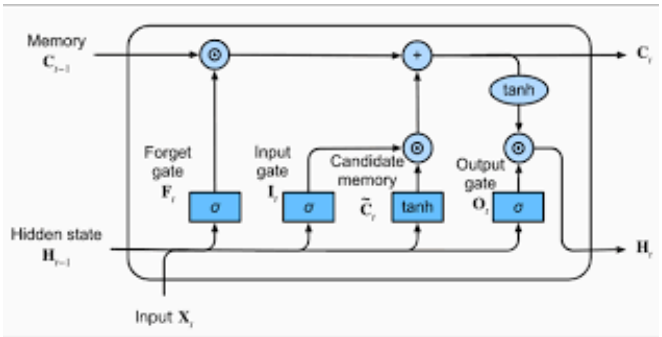

Figura 3.7: Architettura LSTM (da [8])

- L'**output gate** regola l'estrazione e l'utilizzo delle informazioni memorizzate per la fase successiva dell'elaborazione.

La quarta e ultima FNN, chiamata **candidate memory**, ha il compito di generare nuove informazioni candidate da inserire nella memoria. Questa rete produce una possibile nuova rappresentazione delle informazioni da memorizzare, che sarà poi filtrata e selezionata attraverso l'input gate prima di essere effettivamente integrata nella memoria dell'unità LSTM.

Questa struttura consente alla rete di mantenere in memoria informazioni rilevanti per lunghi periodi, permettendo loro di gestire in modo efficace dipendenze a lungo termine, rendendole particolarmente adatte per compiti come l'elaborazione del linguaggio naturale e le serie temporali.

L'unità LSTM riceve in input tre vettori: due di questi provengono dall'unità stessa all'istante precedente (t-1) e sono lo **cell state** (C_{t-1}) e l'**hidden state** (H_{t-1}), mentre il terzo vettore è l'**input** corrente (X_t) ricevuto all'istante t. I **gate** all'interno dell'unità agiscono come **selettori di informazioni**. Ogni gate riceve in input la concatenazione del vettore X_t e dell'hidden state dell'istante precedente (H_{t-1}). In particolare, il forget gate determina quali informazioni eliminare dalla memoria (C_{t-1}). L'input gate, in combinazione con la candidate memory, crea un nuovo vettore, chiamato candidate vector, che viene aggiunto alla memoria, aggiornando così lo cell state a (C_t). Infine, l'output gate decide il valore dell'hidden state all'istante t, che verrà utilizzato dalla successiva unità LSTM all'istante t+1.

Grazie a queste caratteristiche, i sistemi LSTM sono ampiamente utilizzati in applicazioni come il riconoscimento dell'attività umana, ad esempio in dispositivi wearable come Fitbit e Apple Watch, dove elaborano dati sequenziali per monitorare movimenti e attività fisiche. Inoltre, le LSTM trovano impiego nella diagnosi medica[22], dove l'analisi temporale dei dati clinici consente di prevedere l'insorgenza di malattie, migliorando così la capacità di intervento preventivo e il monitoraggio continuo della salute.

3.1.7 Gated Recurrent Unit (GRU)

La **Gated Recurrent Unit** (GRU)[9] è un'architettura di RNN simile alla LSTM, ma con una struttura semplificata che la rende più efficiente dal punto di vista computazionale. La principale differenza tra GRU e LSTM riguarda il modo in cui viene gestito lo cell state. Nella GRU, infatti, lo cell state viene sostituito da un **candidate activation vector**, che viene aggiornato tramite l'azione di due gate: il **reset gate** e l'**update gate**. Questa semplificazione riduce il numero di parametri e rende la GRU più veloce, pur mantenendo la capacità di catturare dipendenze a lungo termine nei dati sequenziali.

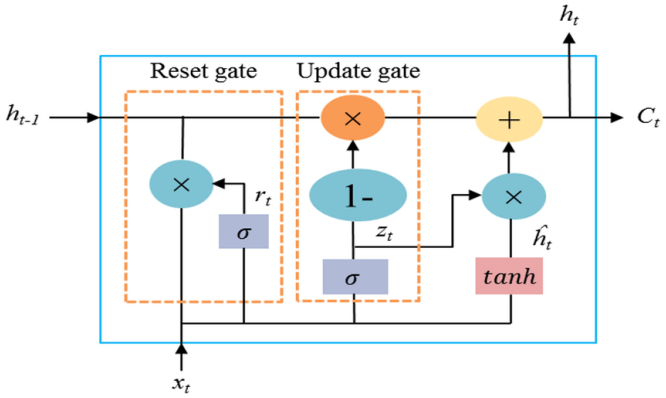

Figura 3.8: Architettura GRU (da [25])

Nella figura **3.8** vediamo la composizione dei gate e il loro meccanismo. Il reset gate determina la porzione di H_{t-1} da dimenticare, per la creazione del nuovo hidden state, mentre l'update gate stabilisce la percentuale di candidate activation vector incorporare nel nuovo hidden state H_t.

Come abbiamo visto GRU e LSTM hanno architetture molto simili, per questo le loro applicazioni sono le medesime, ottenendo prestazioni analoghe, risultando in molti task interscambiabili. Ci sono però contesti in cui una soluzione è preferibile rispetto all'altra. Nel caso in cui le risorse computazionali siano limitate, il modello GRU risulta ottimale data la sua struttura semplificata rispetto a LSTM e quindi dotato di meno parametri addestrabili. In task complessi dove è richiesto di modellare dipendenze a lungo termine, è preferibile un modello più complesso e quindi l'utilizzo di LSTM risulta (a discapito del costo) più efficace.

3.2 Transformers

Le Recurrent Neural Networks (e varianti) viste in precedenza, sono state per lungo tempo considerate all'avanguardia nel campo dell'elaborazione del linguaggio naturale grazie alla loro capacità di gestire dati sequenziali e di catturare dipendenze temporali. Tuttavia, con l'evoluzione del campo, nuove architetture, come i modelli **Transformer** e i modelli basati su di essi, come **BERT** e **GPT**, hanno progressivamente sostituito le RNN come lo stato dell'arte per molte applicazioni NLP. I modelli Transformer, grazie alla loro capacità di catturare dipendenze a lungo termine senza incorrere nei problemi di vanishing gradient, hanno dimostrato di essere più efficaci per compiti complessi come la traduzione automatica, il riassunto testuale e la comprensione contestuale.

3.2.1 Introduzione

I Transformer sono un'architettura innovativa nel campo del Natural Language Processing (NLP), introdotta da Vaswani et al. nel 2017 con il famoso paper "**Attention is All You Need**"[34]. A differenza dei modelli precedenti, come le Recurrent Neural Networks (RNN) o le Long Short-Term Memory (LSTM), i Transformer non elaborano i dati sequenzialmente, ma sfruttano il meccanismo di **self-attention** per gestire relazioni a lungo raggio tra i dati, migliorando l'efficienza e la capacità di parallelizzazione.

Il modello è composto da quattro blocchi principali:

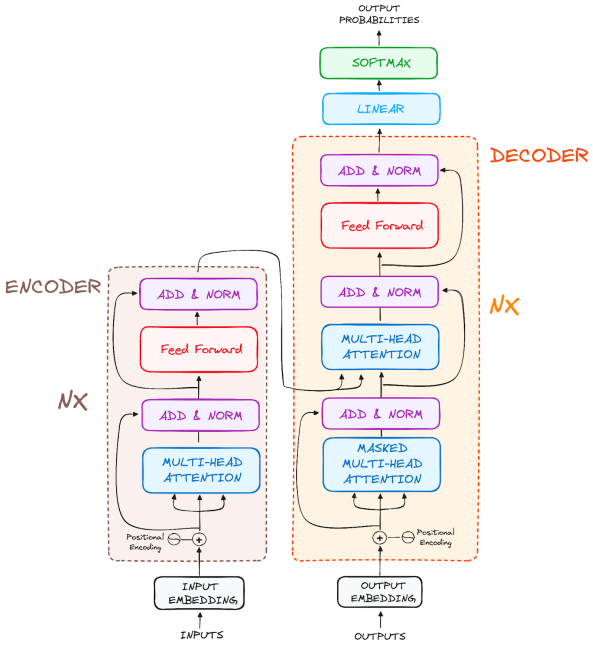

Figura 3.9: Architettura Transformer (da [14])

- **Tokenization**: Il testo in input viene suddiviso in unità chiamate token (parole, frammenti di parole o caratteri), che rappresentano le unità base su cui il modello lavorerà. Ogni token viene quindi associato a un indice numerico per essere processato dalla rete neurale.
- **Embedding**: I token vengono convertiti in vettori numerici mediante un processo di embedding, che rappresenta ciascun token in uno spazio continuo. Questo consente al modello di gestire meglio le relazioni semantiche tra le parole, poiché parole simili avranno vettori simili nello spazio vettoriale.
- **Positional Encoding**: Poiché il Transformer non elabora i dati in sequenza, si aggiunge un positional encoding ai vettori di embedding per fornire al modello informazioni sull'**ordine delle parole**. Questo permette al modello di comprendere la struttura della frase e la posizione relativa dei token.
- **Blocchi Transformer**: L'input arricchito con le informazioni posizionali passa attraverso una serie di blocchi Transformer, composti principalmente da self-attention e feedforward layers. Questi blocchi permettono di apprendere relazioni complesse tra i token, indipendentemente dalla loro distanza nella sequenza.

Infine, per i compiti di classificazione o generazione, l'output passa attraverso una funzione softmax, che restituisce un vettore di probabilità. Questo vettore rappresenta la probabilità che ciascun token o parola appartenga a una determinata classe o sia il prossimo nella sequenza, a seconda del task.

3.2.2 Architettura Encoder-Decoder

L'architettura Transformer è composta da due componenti principali: un **encoder** e un **decoder** (vedi figura 3.9), che lavorano insieme per elaborare input e generare output. Il blocco encoder è responsabile della trasformazione della sequenza di input in una rappresentazione interna ricca di informazioni, mentre

il blocco decoder utilizza questa rappresentazione per produrre l'output, come nel caso della traduzione automatica.

Ogni encoder è formato da una serie di strati che includono il meccanismo di self-attention, e reti feedforward. Il meccanismo di self-attention consente a ogni parola della sequenza di pesare l'importanza delle altre parole nel contesto, indipendentemente dalla loro posizione nella frase. Questo è reso possibile grazie all'uso dei vettori **Query**, **Key** e **Value**, che calcolano la rilevanza tra i token. Successivamente, l'output passa attraverso una rete **feedforward** per applicare operazioni non lineari e migliorare la capacità del modello di apprendere pattern complessi.

Il decoder ha una struttura simile all'encoder, con la differenza che include un meccanismo di **masked self-attention** per garantire che il modello generi l'output in modo sequenziale, senza "guardare" token futuri. Entrambi i blocchi, encoder e decoder, sono dotati di **multi-head attention**, che permette di calcolare l'attenzione in parallelo su diverse parti dell'input, migliorando la capacità del modello di cogliere relazioni complesse.

Infine, sia l'encoder che il decoder si avvalgono del positional encoding, che fornisce informazioni sulla posizione dei token, necessarie per gestire l'ordine della sequenza. L'output finale viene processato tramite una funzione softmax, che restituisce un vettore di probabilità, indicando la parola successiva o la classe predetta. Questa architettura permette di gestire compiti complessi come la traduzione, la generazione di testo e la sintesi del linguaggio.

3.2.3 Attention is All You Need

Il meccanismo di **self-attention** è il cuore dell'architettura Transformer e permette al modello di focalizzarsi sulle parti rilevanti di una sequenza di input durante l'elaborazione, indipendentemente dalla loro distanza nel testo. A differenza dei modelli sequenziali tradizionali come le RNN, che processano le informazioni una alla volta, la self-attention consente di confrontare ogni parola con tutte le altre della sequenza contemporaneamente, catturando le relazioni contestuali tra le parole in modo più efficiente.

Il processo di self-attention si basa su tre vettori: Query (Q), Key (K) e Value (V), che vengono calcolati per ogni parola (o token) dell'input. Il Query rappresenta la parola per cui si vuole capire quali altre parole sono rilevanti, il Key rappresenta ogni altra parola nella sequenza, e il Value contiene le informazioni utili che verranno aggregate.

Per calcolare l'attention, il Query di una parola viene confrontato con i Key di tutte le altre parole nella sequenza, utilizzando un prodotto scalare. Questo confronto produce dei punteggi di **similarità**, che vengono normalizzati tramite softmax per ottenere un insieme di pesi. Questi pesi determinano quanto ogni parola della sequenza contribuisce all'output finale. Infine, i Value associati a ciascuna parola vengono moltiplicati per i rispettivi pesi, e i risultati vengono sommati per ottenere una rappresentazione combinata, in cui le informazioni più rilevanti vengono enfatizzate. La formula matematica per il calcolo dell'attention è il seguente:

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

dove $\sqrt{d_k}$ è la dimensione del vettore Q e K.

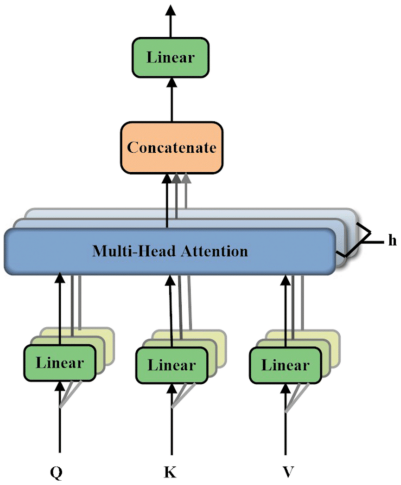

Figura 3.10: Rappresentazione grafica del meccanismo di multi-head attention (da [23])

Questo meccanismo consente al modello di comprendere rapidamente quali parti della sequenza sono più importanti nel contesto, catturando relazioni sia a breve che a lungo termine tra i token, senza la necessità di elaborare i dati in modo sequenziale. Il multi-head attention, una versione parallela della self-attention, viene utilizzato per migliorare ulteriormente la capacità del modello di cogliere molteplici aspetti delle relazioni tra i token.

3.2.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) è un modello di Natural Language Processing (NLP) basato sull'architettura Transformer, sviluppato da Google nel 2018[13]. La principale innovazione portata dall'architettura è la sua capacità di comprendere il contesto delle parole sia guardando a sinistra che a destra della parola in questione, grazie all'uso di un'**architettura bidirezionale**. A differenza dei modelli precedenti, che elaboravano le sequenze in modo unidirezionale (da sinistra a destra o da destra a sinistra), BERT utilizza entrambi i contesti, rendendolo particolarmente efficace nella comprensione del linguaggio naturale.

Il modello viene pre-addestrato su un vasto corpus di testi (come Wikipedia e il BooksCorpus) utilizzando due compiti principali:

- **Masked Language Modeling** (MLM): Alcune parole nella sequenza di input vengono mascherate, e il modello deve predire la parola mancante basandosi sul contesto bidirezionale.
- **Next Sentence Prediction** (NSP): Il modello deve determinare se una frase segue logicamente un'altra, migliorando la sua capacità di gestire relazioni tra frasi.

BERT può essere **fine-tuned** per una varietà di task NLP specifici, come la classificazione del testo, il named entity recognition (NER) o la domanda-risposta, ottenendo prestazioni all'avanguardia. La capacità del sistema di comprendere meglio il contesto ha portato a miglioramenti significativi in numerose applicazioni di NLP, ed è stato un punto di svolta per l'evoluzione dei modelli linguistici. Per queste sue caratteristiche in termini di prestazioni è il modello di riferimento per il nostro task, nonché modello dello stato dell'arte in campo del NLP in cui è fondamentale una comprensione profonda del contesto.

4

Analisi comparativa

In questo capitolo confronteremo i modelli NLP, descritti ampiamente nel capitolo precedente, applicandoli al nostro specifico task. In particolare, forniremo una breve descrizione del dataset utilizzato per l'addestramento e l'inferenza, e presenteremo un'analisi completa delle prestazioni di ciascun modello. L'obiettivo è determinare quale modello risulti il più adatto a risolvere il problema descritto nel capitolo 2, eleggendo infine il modello migliore in termini di accuratezza.

4.1 Il Dataset

Per il confronto tra i modelli NLP, abbiamo utilizzato un **dataset** specifico per il nostro problema, proveniente dall'Università Milano-Bicocca, contenente circa 400.000 referti medici in lingua italiana, ciascuno associato al relativo codice SNOMED-3. Il dataset è stato suddiviso in due parti: un **train set** contenente l'80% dei dati e un **test set** con il rimanente 20%.

I **referti medici** rappresentano una tipologia di testo particolarmente complessa. Ogni parola ha un elevato impatto semantico, e i termini tecnici del linguaggio medico sono spesso intricati, ma cruciali per il contesto clinico. L'accurata comprensione e classificazione di tali termini è essenziale per mantenere la coerenza e la precisione dei referti. Inoltre, la lingua italiana, notoriamente complessa per la sua morfologia e sintassi, aggiunge un ulteriore livello di difficoltà.

Queste caratteristiche rendono la classificazione del testo una sfida significativa, poiché il contesto e la corretta interpretazione delle parole tecniche sono fondamentali per il successo dell'analisi. Pertanto, l'utilizzo di modelli NLP avanzati è cruciale per ottenere prestazioni soddisfacenti in questo scenario.

4.2 Confronto

In questa sezione discuteremo i risultati ottenuti nella classificazione del testo utilizzando i modelli NLP applicati al nostro dataset di referti medici. Verranno esaminate le prestazioni in termini di **accuratezza**, precisione, **recall** e **F1 score**, con l'obiettivo di identificare il modello più adatto a risolvere il problema di classificazione nel contesto medico descritto in precedenza.

Come evidenziato nella tabella 4.1, le baseline hanno ottenuto prestazioni simili tra loro, con le eccezioni di **SVM** e **Logistic Regression**, i cui risultati sono inferiori rispetto agli altri modelli. Que-

Modello	weighted precision	weighted recall	weighted f1-score	accuracy
CNN	0.459	0.474	0.452	0.474
RNN	0.407	0.426	0.401	0.426
LSTM	0.427	0.467	0.427	0.467
GRU	0.427	0.467	0.427	0.467
SVM	0.227	0.368	0.201	0.368
Logistic Regression	0.199	0.374	0.254	0.374
Random Forest	0.482	0.575	0.503	0.575
BERT	0.786	0.786	0.756	0.786

Tabella 4.1: Confronto dei modelli in termini di precision, recall e f1-score

sto comportamento era atteso, dato che entrambi sono modelli relativamente semplici e meno adatti a gestire la complessità semantica e tecnica del nostro task. Al contrario, il modello **BERT** si distingue nettamente, registrando prestazioni eccellenti che superano di gran lunga la soglia delle baseline, confermando le nostre previsioni iniziali. Questa sua eccellenza prestazionale la riscontriamo anche se andiamo a visionare i livelli di accuracy raggiunti, dove BERT supera di significativamente i concorrenti.

Dalla tabella emergono chiaramente i problemi di **scalabilità** già discussi nel capitolo precedente per i modelli Support Vector Machines e Logistic Regression, che hanno registrato i risultati peggiori. Questo limite si è manifestato anche nella fase di addestramento del modello **Random Forest**, dove l'elevato numero di alberi decisionali ha causato alcune difficoltà operative. Nonostante ciò, Random Forest ha ottenuto metriche di prestazione eccellenti, rendendolo una valida alternativa, soprattutto in contesti con una quantità limitata di dati. Per quanto riguarda le **reti neurali**, i modelli hanno mo-

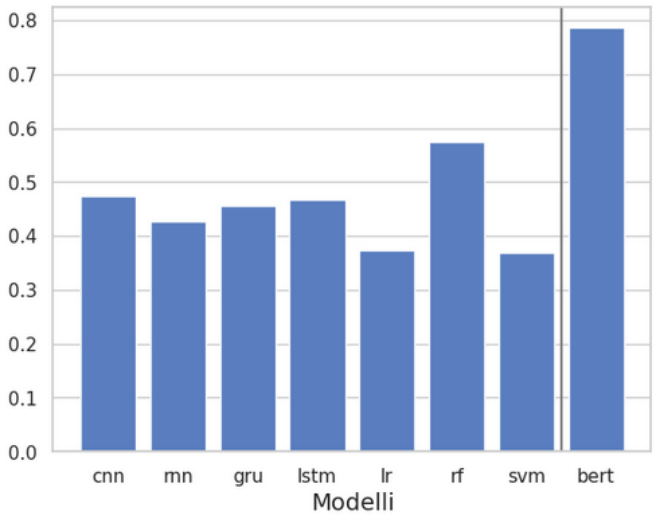

Figura 4.1: Accuracy dei modelli nella classificazione dei referti medici

strato prestazioni molto simili, con metriche quasi identiche in tutte e quattro le valutazioni, nonostante l'implementazione adottata sia piuttosto basilare. Sebbene un aumento della complessità dell'architettura possa potenzialmente migliorare i risultati, ciò esula dallo scopo della nostra analisi. Le reti neurali rimangono comunque una valida opzione per task NLP meno complessi, dove non è necessaria un'architettura sofisticata come i modelli Transformer.

Il modello **BERT** ha dimostrato un'eccellente capacità di assimilare il complesso lessico medico, riuscendo a comprendere efficacemente le relazioni tra il gergo tecnico e le etichette **SNOMED-3**,

cogliendo la **correlazione semantica** tra i termini. Questo lo rende il modello più performante, come confermato dai dati, per task complessi come quello affrontato nel nostro studio. In particolare, il modello è stato **fine-tuned** per adattarlo in maniera specifica al nostro problema, migliorando ulteriormente la sua capacità di interpretare e classificare correttamente i referti medici.

5

Conclusioni

In questa tesi abbiamo esaminato i concetti fondamentali del **Natural Language Processing** (NLP), concentrandoci sulle metodologie chiave, le sfide e le implementazioni pratiche nel contesto della digitalizzazione in medicina. Abbiamo identificato come l'applicazione di tecniche di NLP possa contribuire alla trasformazione digitale del **settore sanitario**, con ripercussioni potenzialmente positive sulla gestione della salute dei pazienti.

L'obiettivo iniziale di questa ricerca era la creazione di uno strumento pratico in grado di affrontare una problematica specifica all'interno del contesto medico, ossia la **classificazione automatica di referti medici**. A tale scopo, abbiamo sviluppato un modello basato su **BERT** (Bidirectional Encoder Representations from Transformers), addestrato per comprendere e classificare referti medici in modo sufficientemente accurato. Questo sistema può fornire supporto nella redazione e nella prima classificazione automatica dei referti, agevolando così il lavoro degli operatori sanitari.

Il modello sviluppato rappresenta un contributo alla crescente necessità di strumenti automatizzati nel settore sanitario, dove il lavoro manuale predomina ancora. Sebbene il modello proposto non rappresenti una soluzione completa, esso dimostra come l'**NLP** possa essere applicato per **migliorare l'efficienza e la precisione** nella gestione dei dati medici. L'adozione di tecnologie di questo tipo potrebbe ridurre il carico di lavoro degli operatori sanitari, consentendo loro di concentrarsi su compiti più complessi e migliorando l'efficienza operativa.

Dal punto di vista pratico, anche i cittadini possono trarre vantaggio dall'uso di strumenti più efficaci e precisi per monitorare e gestire la propria salute, accedendo a servizi medici migliorati grazie alla digitalizzazione.

Il modello BERT implementato in questo progetto, sebbene funzionale, presenta alcuni limiti. Uno dei principali è legato all'addestramento su un **dataset non bilanciato**, che ha limitato l'accuratezza della classificazione. Inoltre, esistono modelli di base più avanzati che potrebbero fornire prestazioni superiori rispetto a BERT. Un altro limite è la classificazione a singola etichetta, che non sfrutta appieno la **struttura gerarchica** di codifiche mediche come SNOMED-3.

In futuro, la ricerca potrebbe essere ampliata in diverse direzioni. In primo luogo, un miglioramento potrebbe essere ottenuto attraverso l'addestramento su un dataset bilanciato, il che aumenterebbe l'affidabilità del modello. Inoltre, l'implementazione di un modello di **classificazione multi-label** permetterebbe di sfruttare la struttura gerarchica di SNOMED-3, offrendo etichette più specifiche e

accurate. Infine, potrebbe essere utile esplorare l'uso di modelli più avanzati come **T5** o **GPT-4**, che potrebbero migliorare le prestazioni complessive.

Nonostante le limitazioni, il lavoro svolto rappresenta un piccolo ma significativo passo verso la digitalizzazione di un settore ancora fortemente manuale come quello sanitario. Strumenti come il modello BERT proposto in questa tesi possono fornire un supporto tangibile in termini di efficienza ed efficacia, riducendo il carico di lavoro umano. In definitiva, la **digitalizzazione del settore medico**, se ben implementata, potrebbe portare benefici sia agli operatori sanitari che ai pazienti, migliorando la qualità e l'accessibilità delle cure.

Bibliografia

[1] Md. Zahangir Alam, M. Saifur Rahman, e M. Sohel Rahman. A random forest based predictor for medical data classification using feature ranking. *Informatics in Medicine Unlocked Volume 15, 100180*, 2019.

[2] Talal Almutiri e Farrukh Nadeem. Markov models applications in natural language processing: A survey. *International Journal of Information Technology and Computer Science (IJITCS) IJITCS Vol. 14, No. 2*, 2022.

[3] Ahmad Alsaleh e Cahit Perkgoz. A space and time efficient convolutional neural network for age group estimation from facial images. *PeerJ Computer Science 9:e1395*, 2023.

[4] Mochammad Sahal andZulkifli Hidayat, Resqi Abdurrazzaaq Putra, e Muhammad Azriel Rizqifadiilah andFirdaus Dheo Saputra. Lane keeping system using convolutional neural network for autonomous car. *14th International Conference on Information & Communication Technology and System (ICTS)*, 2023.

[5] Niraj Bhoi. Stemming vs lemmatization in nlp. *Medium*, 2022.

[6] Leo Breiman. Random forests. *Machine Learning 45, 5–32*, 2001.

[7] Danny Britz. Understanding convolutional neural networks for nlp. *Danny's Blog*, 2015.

[8] Ottavio Calzone. An intuitive explanation of lstm. *Medium*, 2022.

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, e Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv Prepr. arXiv:1406.1078*, 2014.

[10] Corrina Cortes e Vladimit Vapnik. Support-vector networks. *Machine Learning 20, 273–297*, 1995.

[11] J.S. Cramer. The origins of logistic regression. *Tinbergen Institute Working Paper No. 2002-119/4*, 2002.

[12] Khurana D., Koli A., e Khatter K. et al. Natural language processing: state of the art, current trends and challenges. *Multimed Tools Appl 82, 3713–3744*, 2022.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, e Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv Prepr. arXiv:1810.04805*, 2018.

[14] Josep Ferrer. How transformers work: A detailed exploration of transformer architecture. *DataCamp*, 2024.

[15] Jan Niklas Fuhg, A. Karmarkar, Teeratorn Kadeethum, Hongkyu Yoon, e N. Bouklas. Deep convolutional ritz method: parametric pde surrogates without labeled data. *Applied Mathematics and Mechanics 44(7):1151-1174*, 2023.

[16] Sepp Hochreiter e Jurgen Schmidhuber. Long short-term memor. *Neural Computation 9(8):1735-1780*, 1997.

[17] Chih-Wei Hsu e Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks (Volume: 13, Issue: 2)*, 2002.

[18] Logistic Regression in Medical Research. Patrick schober and thomas r. vetter. *Anesthesia & Analgesia 132(2):p 365-366*, 2021.

[19] Arya Karn. Top applications of natural language processing (nlp) in 2024. *Intellipaat*, 2024.

[20] Mr Farkhod Khushaktov. Introduction random forest classification by example. *Medium*, 2023.

[21] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv Prepr. arXiv:1408.5882*, 2014.

[22] Zachary C. Lipton, David C. Kale, Charles Elkan, e Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv Prepr. arXiv:1511.03677*, 2015.

[23] Xiao Luwei, Xiaohui Hu, Yinong Chen, Yun Xue, Bingliang Chen, Donghong Gu, e Bixia Tang. Multi-head self-attention based gated graph convolutional networks for aspect-based sentiment classification. *Multimedia Tools and Applications 81(1–2):1-20*, 2022.

[24] Domenico Marino. Fascicolo sanitario elettronico, ecco tutti i problemi e come risolverli. *Agenda Digitale*, 2019.

[25] Saeed Mohsen. Recognition of human activity using gru deep learning algorithm. *Multimedia Tools and Applications 82(30)*, 2023.

[26] Keiron O'Shea e Ryan Nash. An introduction to convolutional neural networks. *arXiv Prepr. arXiv:1511.08458*, 2015.

[27] Anna Francesca Pattaro. Fascicolo sanitario elettronico, cos'è, a che serve e come attivarlo. *Agenda Digitale*, 2024.

[28] Sushmita Poudel. Recurrent neural network (rnn) architecture explained. *Medium*, 2023.

[29] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, e Xuanjing Huang. Pre-trained models for natural language processing: A survey. *arXiv Prepr. arXiv:2003.08271*, 2020.

[30] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Wei Li Yanqi Zhou, e Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv Prepr. arXiv:1910.10683*, 2019.

[31] Olaf Ronneberger, Philipp Fischer, e Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv Prepr. arXiv:1505.04597*, 2015.

[32] Sara Scarpinati. Medicina preventiva e l'utilizzo dell'intelligenza artificiale. *Digital Healt Italia*, 2021.

[33] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv Prepr. arXiv:1912.05911*, 2019.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, e Illia Polosukhin. Attention is all you need. *arXiv Prepr. arXiv:1706.03762v7*, 2017.

[35] I-Tung Yang e Handy Prayogo. Efficient reliability analysis of structures using symbiotic organisms search-based active learning support vector machine. *Buildings 12(4):455*, 2022.

[36] Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V. Vasilakos, e Thippa Reddy Gadekallu. Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *arXiv Prepr. arXiv:2305.10435*, 2023.