Università degli Studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Magistrale in
Scienze Statistiche

Confronto tra modelli linguistici e approcci tradizionali per l'attribuzione d'autore: un'analisi di testi prodotti da studenti e da ChatGPT

Relatore Prof. Andrea Sciandra
Correlatore Prof. Michele A. Cortelazzo

Laureando Francesco Dal Cero
Matricola N. 2091004

Anno Accademico 2023/2024

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

How many questions does it usually take to spot one?
Scott, R. (Director). (1982). *Blade Runner*.
The Ladd Company, Shaw Brothers, Warner Bros.

Con questa domanda rivolta a Harrison Ford nei panni dell'agente Rick Deckard nel celebre film di Ridley Scott si apre una tematica particolarmente discussa negli ultimi anni e sempre più rilevante: è possibile distinguere un umano da un computer? Sebbene la risposta sia sempre stata affermativa e spesso scontata, i recenti sviluppi tecnologici hanno messo in discussione la questione, scatenando polemiche e costringendo le persone a riflettere sull'ambiguità del progresso e sulle sue conseguenze. Il tema della prevaricazione delle macchine sull'uomo è sempre stato relegato a scopi cinematografici (Kubrick, 1968) o riservato a libri di fantascienza, sebbene con la consapevolezza di stare parlando di un futuro non troppo lontano. Grazie al rapido avanzamento tecnologico e informatico che ha caratterizzato gli ultimi decenni, queste tematiche sono divenute ormai attualità e il 2022 potrebbe aver segnato un punto di non ritorno.

Nel novembre di quell'anno il laboratorio di ricerca OpenAI ha rilasciato *ChatGPT*, un *chatbot* basato sull'intelligenza artificiale e sull'apprendimento automatico. Modelli generativi di questo tipo hanno attirato l'attenzione negli ultimi anni per la loro capacità di produrre testi come se fossero stati scritti da umani. La precisione nelle risposte e l'accuratezza dei particolari hanno reso ormai pressoché impossibile distinguere, senza l'utilizzo di strumenti informatici, un testo scritto da una persona da uno generato da un computer.

In questa tesi si vuole trattare il tema del riconoscimento automatico dei testi per distinguerli a seconda che essi siano stati originati da un umano o da un *Large Language Model* (LLM[1]). Il riconoscimento d'autore è una tematica che ha però radici molto più antiche: esse risalgono addirittura al IV secolo con la Donazione di Costantino, un do-

[1] Questo termine verrà approfondito nel Capitolo 1.

cumento storico poi dimostrato falso dal filologo italiano Lorenzo Valla (Maffei, 1980). Esempi più recenti sono noti anche nella letteratura novecentesca che vede protagonista il quindicesimo libro della serie "Il Mago di Oz" il cui autore è stato a lungo incerto (Binongo, 2003) e in quella contemporanea. Un caso riguarda l'autrice Elena Ferrante (Cortellazzo e Tuzzi, 2018), noto pseudonimo per mascherare un'identità segreta, o la più nota J. K. Rowling che ha pubblicato diversi volumi sotto il nome di Robert Galbraith. Un dibattito ancora aperto riguarda la produzione del celebre romanzo "Il buio oltre la siepe", pubblicato nel 1960 e vincitore del Premio Pulitzer per la letteratura nel 1961, ufficialmente riconosciuta a Harper Lee (Shields, 2006) ma forse appartenente al suo amico d'infanzia Truman Capote. L'attribuzione di un testo ad un particolare scrittore è una pratica che ha sollevato questioni per secoli e che ha riscontrato l'interesse soprattutto degli umanisti ma anche di figure appartenenti ad ambiti come giornalismo e politica. L'avvento dei computer e il rapido progresso informatico hanno permesso di disporre di una vasta quantità di corpus[2], rendendo più agevole ed efficiente questa disciplina che può ora usufruire delle più recenti tecnologie e metodologie statistiche.

Scoperte e innovazioni hanno, in passato, suscitato spesso diffidenza, necessitando di tempo per essere accettate e diventare di uso comune. Così parlava, riguardo alla scrittura, il Socrate di Platone nel Fedro (370 a.C.): "*La scrittura è disumana, poiché finge di ricreare al di fuori della mente ciò che in realtà può esistere solo al suo interno*". L'accusa principale è che la scrittura privasse l'uomo del proprio pensiero, della capacità di creare indebolendo la mente. È normale quindi, che ora, di fronte a un cambiamento così grande ci si interroghi sulla capacità dei computer di produrre un testo indistinguibile da uno di provenienza umana, riflettendo non solo sulle conseguenze pratiche ma anche su quelle ontologiche ed epistemologiche di questa intersezione tra creatività umana e capacità computazionali. La possibilità di simulare il pensiero umano è stata considerata a lungo utopistica e pretenziosa, accessibile in parte con la nascita delle reti neurali ma solo ora divenuta realtà attraverso la riproduzione artificiale di testi.

Se l'impronta digitale di una persona può rivelare istantaneamente la sua identità, lo stile di scrittura di un autore ha una funzione quasi analoga: difficilmente distinguibile ad occhio, contribuisce a differenziare la produzione artistica di uno scrittore rendendola unica e inimitabile. Ci sono buone ragioni per credere che le persone abbiano una propria *authorial fingerprint* rilevabile nei loro scritti: un insieme di tratti e caratteristiche di cui si può usufruire per identificarne univocamente l'artefice. Viene naturale chiedersi se questo valga anche per testi prodotti da un'intelligenza artificiale, o più propriamente da un modello linguistico. Generalizzando la questione, si può estendere il problema

[2]Raccolta ordinata e completa di opere o di autori.

domandandosi se gli uomini e i computer abbiano stili di scrittura riconoscibili e diversi tra loro. Al di là della mera capacità di generare testi coerenti, si aprono interrogativi cruciali riguardo alla distinzione tra l'autenticità della scrittura umana e quella prodotta artificialmente. La creatività, l'originalità, l'imprevedibilità che definiscono un uomo possono essere riprodotte da una macchina? Diventa dunque indispensabile disporre di strumenti e tecniche capaci di distinguere testi di entrambi i tipi, che vadano oltre il banale criterio di presenza o assenza di termini.

Questo lavoro nasce anche dalla lettura di un articolo sul sito di OpenAI[3] in cui la stessa azienda sosteneva che il suo classificatore *AI Text Classifier* non fosse in grado di rilevare accuratamente se un testo fosse stato prodotto dall'intelligenza artificiale o meno, rendendolo momentaneamente non disponibile.

L'obiettivo di questa tesi è verificare se sia possibile distinguere accuratamente i testi prodotti da un modello linguistico da quelli scritti dalle persone, prendendo in considerazione un ampio assortimento di metodi e tecniche alla ricerca dei più efficaci.

La tesi è strutturata come segue. Il Capitolo 1 contestualizza l'ambito di ricerca dell'attribuzione d'autore, per poi introdurre importanti quantità di analisi come i trasformatori e i rilevatori automatici. Nel Capitolo 2 viene descritta la creazione del corpus, così come i dati e le metodologie a disposizione; successivamente vengono proposte alcune analisi esplorative volte a fornire uno sguardo complessivo delle relazioni tra i testi. Il Capitolo 3 è dedicato allo sviluppo e al perfezionamento dei modelli di classificazione, a cui segue una approfondita interpretazione dei risultati. Il Capitolo 4, dal taglio più descrittivo, permette di confrontare quanto conseguito nei capitoli precedenti con ciò che è possibile ottenere attraverso gli strumenti disponibili in rete, per poi offrire alcuni spunti di riflessione di carattere etico e metodologico. Infine, al termine delle Conclusioni è consultabile un'Appendice dove vengono descritti nel dettaglio alcuni dei modelli adoperati per la realizzazione del progetto.

Le analisi svolte in questo lavoro sono state eseguite usando il *software* statistico `R` (v4.2.3; R-Core-Team, 2024) e il linguaggio di programmazione `Python` (v3.10.7; Python Reference Manual, 1995).

[3] https://openai.com/.

Capitolo 1

Il contesto dell'attribuzione d'autore

1.1 Cenni su storia e sviluppo del riconoscimento d'autore e dell'elaborazione del linguaggio

L'identificazione e l'attribuzione dei testi a specifici autori rappresentano sfide con una lunga storia, costituendo alcuni dei campi più antichi ma allo stesso tempo innovativi nell'ambito dell'*information retrieval*[1] (Burrows *et al.*, 2009, Bozkurt *et al.*, 2007). Le controversie sull'autenticazione di scritti e documenti esistono da quando sono stati creati i primi testi e nel corso dei secoli si sono susseguite numerose questioni al riguardo, molte delle quali ancora irrisolte. Alcuni studiosi hanno sollevato dubbi sull'autenticità di alcune epistole attribuite all'apostolo Paolo nel Nuovo Testamento o, senza andare troppo indietro nel tempo, esistono diversi testi come Edoardo III, che sono stati riconosciuti a William Shakespeare (1564-1616) ma il cui autore è incerto. Per risolvere questi quesiti, le tecniche utilizzate fino agli ultimi decenni del ventesimo secolo riguardavano lo studio dei temi e dei contenuti trattati nei testi, l'analisi del vocabolario, la scelta delle parole, la struttura delle frasi, il tono, l'uso della punteggiatura e altri aspetti stilistici. Questi metodi richiedevano un lavoro manuale e un'esperienza significativa da parte degli studiosi per valutare e confrontare i testi in modo accurato. Sebbene fossero spesso efficaci nel determinare l'attribuzione di un testo incerto, erano anche soggetti a interpretazioni soggettive e potenziali errori.

L'avvento della statistica moderna, con i suoi potenti mezzi e innovativi strumenti, sostenuta dalla comparsa dei primi computer, ha permesso di approfondire queste tematiche attraverso algoritmi e tecniche di estrazione automatica dell'informazione. Parallelamente a ciò, le grandi quantità di dati e di corpus a disposizione hanno fatto sì che l'analisi

[1] Reperimento dell'informazione.

dei testi e l'attribuzione d'autore diventassero aree di ricerca popolari e produttive. Negli anni il significato di "*autorship attribution*" non è cambiato e può essere definito come un qualsiasi tentativo di dedurre il creatore di un generico dato linguistico attraverso le caratteristiche dello stesso (Juola *et al.*, 2006). Questa ampia definizione può comprendere anche applicazioni riguardanti il riconoscimento vocale ma solitamente l'attenzione si concentra sui testi scritti. Molti ricercatori (Juola *et al.*, 2006) sono dell'idea che le persone quando scrivono lascino un'impronta che definisce il loro stile e che possa essere rilevata nei loro testi. Siccome gli individui apprendono le proprie conoscenze e creano il proprio linguaggio in modo differente, le loro produzioni scritte saranno a loro volta diverse secondo alcuni aspetti. Ci sono valide ragioni per pensare che queste impronte esistano, così come altrettanti motivi per ritenere che riconoscerle possa risultare difficile e complesso.

Il linguaggio umano è un sistema estremamente sofisticato e combina un sorprendente numero di regolarità con un alto grado di variabilità. Perché sia analizzato è necessario osservare che può essere considerato come un flusso ordinato di eventi come lettere, parole e frasi, estratti da una popolazione di eventi potenziali. Se si osserva la relazione tra questi eventi si possono riconoscere alcune regolarità che identificano diversi stili di scrittura e auspicabilmente il loro autore.

Anche ammettendo che ciò sia possibile, negli ultimi decenni si è affiancata un'ulteriore sfida nell'ambito del riconoscimento dei testi, dovuta all'avvento del Natural Language Processing (NLP). Con questo termine ci si riferisce a sistemi informatici che analizzano, tentano di comprendere, o di produrre una o più lingue umane (Allen e James, 2003). Questo ambito di ricerca è sorto negli anni '50 e '60, grazie ai primi esperimenti e analisi sul linguaggio naturale e sulla sua elaborazione computazionale. Nell'ottobre del 1950 il matematico britannico Alan Turing ha proposto nel suo articolo, *Computing machinery and intelligence* (1950), un criterio per determinare se una macchina fosse in grado di esibire un comportamento intelligente. Il test presentato si svolge in modo simile ad una conversazione tra un umano e una macchina (generalmente tramite un computer): un interrogatore umano ha una conversazione scritta con i due partecipanti. Non sa quale dei due sia umano e quale sia la macchina. Se l'interrogatore non riesce a distinguere in modo affidabile tra i due interlocutori basandosi sulle risposte, si considera che la macchina abbia superato il test. Se una macchina può convincere un essere umano che sta interagendo con un'altra persona, allora si può ritenere che abbia dimostrato un certo livello di intelligenza artificiale.

Negli anni successivi si sono susseguiti vari avvenimenti chiave nella storia dell'elaborazione del linguaggio: Alan Hodgkin e Andrew Huxley (1952) hanno descritto tramite

un modello matematico il comportamento dei neuroni nel cervello umano; Noam Chomsky ha rivoluzionato il campo linguistico con il suo libro *Syntactic Structures* (1957), suggerendo che la struttura grammaticale delle frasi dovesse essere cambiata e proponendone una nuova, affinché i computer potessero comprendere il linguaggio umano. Questo tipo di grammatica si basa sull'idea che le frasi possono essere analizzate in termini di diverse fasi o livelli di struttura sintattica, ciascuna delle quali contribuisce alla comprensione complessiva della frase. Nel 1958, John McCarthy ha sviluppato il linguaggio di programmazione LISP, in uso ancora oggi, mentre nel 1964 è stato creato ELIZA, un programma che simulava una conversazione con uno psichiatra utilizzando tecniche di riflessione. Tuttavia, non vi era ancora una reale comprensione del linguaggio da parte del computer; la vera rivoluzione è avvenuta alla fine degli anni '80 passando da complesse regole "scritte a mano" ad algoritmi di apprendimento automatico e modelli statistici (come gli alberi decisionali) in grado di prendere decisioni probabilistiche. Negli anni '90 la popolarità di questi metodi per il NLP è cresciuta ulteriormente, vedendo la nascita nel 1997 dei modelli a rete neurale ricorrente (RNN) utilizzati in particolare per l'elaborazione della voce e del testo (Hochreiter *et al.*, 1997). Nel 2001, Yoshua Bengio e il suo gruppo di ricerca hanno proposto il primo modello di linguaggio neurale, utilizzando una rete *feed-forward* (Bengio *et al.*, 2003). Questo tipo di architettura, molto diversa dalle RNN, descrive una rete neurale artificiale che non utilizza connessioni per formare un ciclo: i dati si muovono solo in una direzione, dai nodi di ingresso, attraverso eventuali nodi nascosti, e poi verso i nodi di uscita. Negli anni 2010, l'introduzione delle reti neurali convoluzionali (CNN), ha permesso di ottenere risultati significativi in compiti di NLP come il riconoscimento delle entità, l'analisi del *sentiment* (Mesnil *et al.*, 2014), e la traduzione automatica che hanno consentito ai computer di comprendere e generare il linguaggio umano in modo più efficace e sofisticato.

L'apice di questa evoluzione è rappresentato dall'emergere dei *Large Language Models* (LLM) come BERT (Devlin *et al.*, 2019) e GPT. Questi modelli, introdotti nel 2018, hanno rivoluzionato il campo dell'NLP grazie alla loro capacità di catturare relazioni semantiche complesse in testi di lunghezza arbitraria. Grazie all'uso di trasformatori e addestramento su grandi corpus di testo, i LLM sono in grado di comprendere e generare testo in modo più naturale e contestuale rispetto ai modelli precedenti. Tuttavia, nonostante i continui sviluppi, nessuno strumento o modello ad oggi è stato in grado di superare il test di Turing e non sembra essere in procinto di farlo.

1.2 I Large Language Models

1.2.1 I modelli linguistici

Un modello linguistico consiste essenzialmente nell'uso di varie tecniche statistiche e probabilistiche basate sul lavoro di algoritmi specificamente addestrati per determinare la probabilità che una determinata sequenza di parole si verifichi in una frase. In termini generali, i modelli linguistici utilizzano metriche e regole linguistiche per comprendere la struttura e il significato del linguaggio umano. Modelli di questo tipo analizzano grandi quantità di testi per fornire una base per le loro previsioni e determinano le loro probabilità al fine di prevedere e produrre nuove frasi e testi coerenti. In questo caso si parla di generazione del linguaggio naturale (Natural Language Generation o NLG).

Con linguaggio naturale, o linguaggio ordinario, si intende un qualsiasi linguaggio che si verifica naturalmente in una comunità umana attraverso un processo di uso, ripetizione e cambiamento dei termini senza pianificazione o premeditazione cosciente (McDonald e David, 2010). Di conseguenza con le dovute proporzioni e precisazioni, il processo di funzionamento dei modelli linguistici è in astratto simile a quello con cui gli esseri umani trasformano un pensiero in parola o in scrittura.

Le reti neurali artificiali sono un tentativo di emulare in modo semplificato il funzionamento delle reti neurali biologiche, sfruttando principi di apprendimento e adattamento per risolvere problemi complessi. Le reti sono composte da nodi, organizzati in strati (*layer*) interconnessi; il principio di funzionamento è simile a quello del cervello umano: i nodi sono l'equivalente dei nostri neuroni, mentre i parametri corrispondono alle nostre sinapsi. Di conseguenza, la quantità dei parametri di un modello fornisce una buona approssimazione di quanto sarà accurato il lavoro da esso svolto. A conferma di ciò, i risultati forniti dagli ultimi anni di ricerca (Wei *et al.*, 2022) hanno mostrato che le prestazioni dei modelli linguistici sono in funzione di una legge di potenza delle dimensioni del modello, delle dimensioni del *set* di dati e della quantità di calcolo. Per questo motivo, a determinare la dimensione dei modelli linguistici, definendoli *Large Language Models*, è il numero di parametri, in continua crescita (Figura 1.1), impiegati all'interno delle reti neurali da cui sono costituiti.

I modelli linguistici si impongono dunque come strumento fondamentale nel campo del trattamento automatico, del linguaggio naturale e dell'intelligenza artificiale. La funzione principale per cui sono progettati è comprendere il linguaggio umano e generare testi sensati in risposta a determinati *input*. Per fare ciò analizzano la struttura semantica e sintattica del testo per estrarre significato e contesto. Ma i loro impiego non si limita a

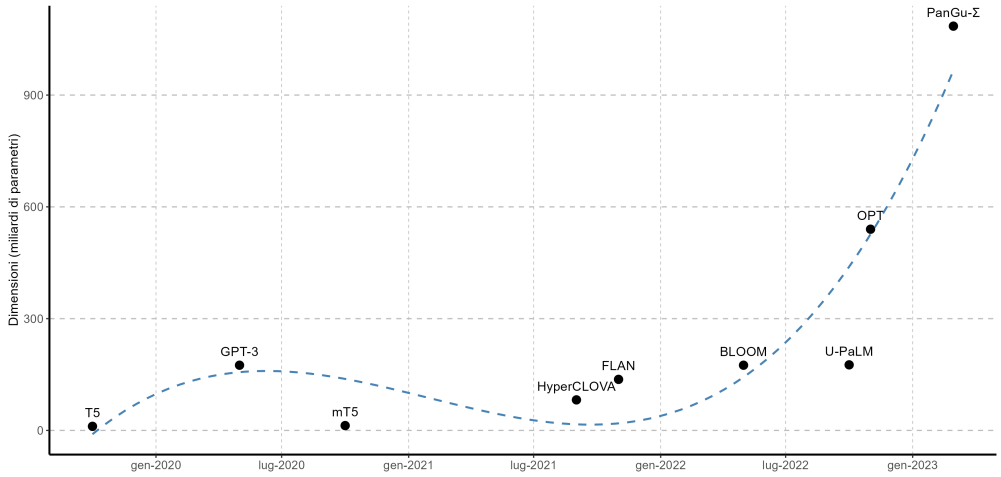


Figura 1.1: Variazione della dimensione dei LLM nel corso degli anni, in termini di numero di parametri al loro interno. Dati ricavati dall'articolo: *A Survey of Large Language Models* (Zhao *et al.*, 2023).

ciò: vengono infatti utilizzati anche negli ambiti di traduzione automatica, sintesi vocale, analisi della polarità, correzione automatica e molto altro.

1.2.2 I trasformatori

Un aspetto fondamentale che ha contribuito allo sviluppo di quest'area di ricerca è l'uso di architetture di trasformatori e del meccanismo di attenzione sottostante (Vaswani *et al.*, 2017), che hanno migliorato notevolmente la capacità dei modelli linguistici di gestire le dipendenze a lungo raggio nei testi in linguaggio naturale (Kasneci *et al.*, 2023). Il meccanismo di attenzione consente al modello di comprendere meglio le relazioni tra le parole di una frase, indipendentemente dalla loro posizione. Per fare ciò il sistema, quando genera previsioni, si concentra sul determinare la rilevanza delle diverse parti dell'*input* proposto al fine di tracciare le dipendenze globali tra *input* e *output*. Questa tecnica chiave nei trasformatori permette al modello di dare maggior peso alle parole rilevanti in una data sequenza di testo durante il processo di apprendimento.

Più nel dettaglio, il modello di attenzione introdotto da Vaswani e colleghi si propone di migliorare il meccanismo precedente che si basava su complesse RNN o CNN che comprendevano un *encoder* (codificatore) e un *decoder* (decodificatore). Il nuovo modello si basa a sua volta sulla struttura *encoder-decoder* ma gli strati sono connessi dal meccanismo di attenzione che permette di rinunciare completamente a reti ricorrenti

e convoluzionali. Inoltre, questa modifica consente al modello di essere più facilmente parallelizzabile e richiede meno tempo di addestramento.

Nello specifico, il codificatore mappa una sequenza in *input* $(x_1, \ldots, x_n)$, in una rappresentazione continua $z = (z_1, \ldots, z_n)$ e dato z il decodificatore genera una sequenza $(y_1, \ldots, y_n)$ in *output* procedendo un elemento alla volta. Il modello è autoregressivo e ad ogni step utilizza l'ultimo *output* come un *input* addizionale per generare il successivo. Il costo computazionale necessario a mantenere l'ordine sequenziale è ridotto a un numero costante di operazioni, a discapito di una riduzione della risoluzione effettiva, contrastata però dal meccanismo di attenzione multipla descritto in seguito.

Il trasformatore utilizza questa struttura generale impilando strati di diversa tipologia come strati *point-wise* e *auto-attenzione* connessi tra loro. Uno strato *point-wise* è un tipo di strato che opera indipendentemente su ciascun elemento della sequenza in ingresso. Questo significa che ogni elemento della sequenza viene elaborato individualmente, senza considerare la relazione con gli altri elementi. Queste due tipologie di strati vengono utilizzati assieme per aggiungere complessità e capacità di modellazione alla rete neurale. Dopo che lo strato di *auto-attenzione* ha catturato le relazioni globali tra gli elementi della sequenza, uno strato *point-wise* può essere applicato separatamente a ciascun elemento per eseguire ulteriori trasformazioni locali.

L'architettura del trasformatore (Figura 1.2) prevede un codificatore costituito da una pila di sei strati identici ognuno composto da due sottolivelli. Il primo è chiamato di *multi-head self attention* ed il secondo è detto *feed-forward Network* (FFN). In particolare, il sottolivello FFN è completamente connesso e consiste in due trasformazioni lineari con una funzione di attivazione $ReLU$ tra di esse, eseguite su ciascuna posizione di una sequenza.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

L'attivazione $ReLU$ è rappresentata da $\max(0, xW_1 + b_1)$, mentre W_1 e W_2 sono le matrici dei pesi delle due trasformazioni lineari.

Anche il decodificatore è composto da una pila di sei strati identici ed oltre ai due sottolivelli in ogni strato, ne è presente un terzo che esegue il meccanismo di *multi-head attention* sull'*output* della pila del codificatore. Il codificatore processa l'*input* (ad esempio, una sequenza di parole) per codificarlo in una rappresentazione appropriata, mentre il decodificatore genera l'*output* (ad esempio, una traduzione) utilizzando questa rappresentazione. In questo modo, per come è costruito il modello, le previsioni per un token in posizione i, dipendono solo dagli *output* noti sulle posizioni precedenti ad i.

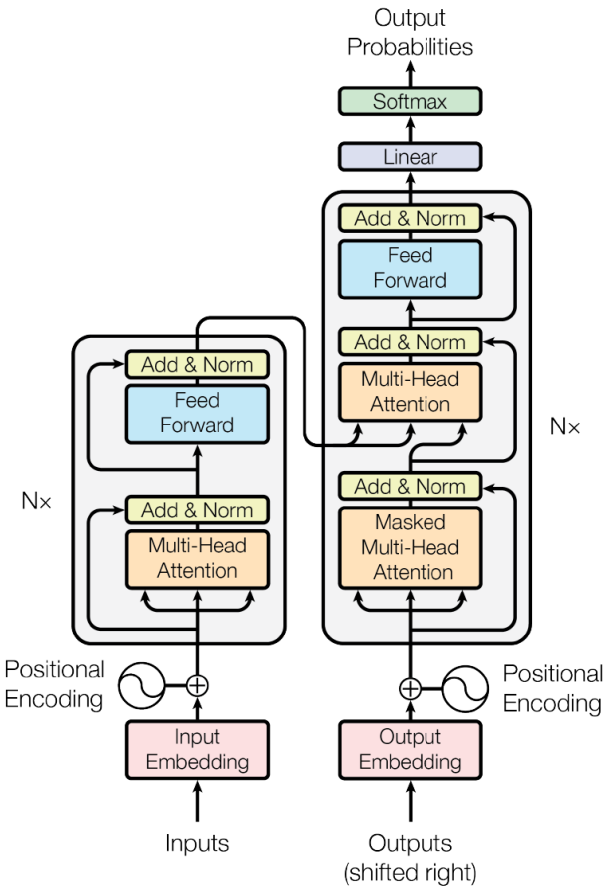


Figura 1.2: Il trasformatore - architettura del modello, tratta da Vaswani *et al.* (2017).

Una *funzione di attenzione* può essere descritta come una mappatura di una *query* (quindi una richiesta) e di un insieme di coppie chiave-valore su un *output*, in cui la *query* Q, le chiavi K, i valori V e l'*output* sono tutti vettori. L'*output* è calcolato come somma ponderata dei valori V, dove il peso assegnato a ciascun valore è calcolato da una funzione di compatibilità della *query* con la chiave corrispondente per indicare quanto ciascun elemento sia rilevante rispetto alla *query*. La funzione serve quindi a calcolare l'importanza relativa di ciascun elemento rispetto agli altri, all'interno di una stessa sequenza di *input*. Questa importanza viene poi utilizzata per pesare l'influenza di ciascun elemento durante la fase di calcolo della rappresentazione complessiva della sequenza. La funzione consente al modello di focalizzarsi su parti specifiche dell'*input* durante il processo di elaborazione, migliorando così la capacità del modello di catturare relazioni complesse e di gestire sequenze di lunghezza variabile.

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

La funzione di compatibilità per il calcolo dei pesi W è una funzione *softmax*, che normalizza i pesi e garantisce che la loro somma per ciascun token sia pari a 1. Le *query* e le chiavi sono di dimensione d_k e il prodotto scalare QK^T viene moltiplicato per $\frac{1}{\sqrt{d_k}}$ in modo che i valori di attenzione non diventino troppo grandi quando la dimensione delle rappresentazioni aumenta.

Invece di eseguire una singola funzione di attenzione con chiavi, valori e interrogazioni con un numero fisso di d_{modello} dimensioni (solitamente pari a 512), il meccanismo di *multi-head attention* permette che vengano proiettate linearmente le *query*, le chiavi e i valori h volte (con h numero di *heads* o strati del modello), tramite proiezioni lineari diverse su dimensioni diverse. Queste dimensioni sono pari a $d_k = \frac{d_{\text{modello}}}{h}$ per *query* e chiavi e $d_v = d_k = \frac{d_{\text{modello}}}{h}$ per i valori. Su ciascuna di queste versioni proiettate di *query*, chiavi e valori si esegue la funzione di attenzione in parallelo, ottenendo valori di uscita d_v-dimensionali. Questi ultimi vengono concatenati e ancora una volta proiettati, ottenendo i valori finali.

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{dove} \quad \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

Le proiezioni sono matrici di parametri $W_i^Q \in \mathbb{R}^{d_{\text{modello}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{modello}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{modello}} \times d_v}$, e $W^O \in \mathbb{R}^{hd_v \times d_{\text{modello}}}$.

Questo meccanismo consente al modello di partecipare congiuntamente alla creazione di informazione proveniente da sottospazi in posizione diverse, diversamente da quanto possibile con un meccanismo di attenzione *single-head*. Usando strati diversi in modo parallelo, la loro dimensione si riduce e il costo computazionale, pur usando più strati, rimane simile a quello di un meccanismo *single-head* con dimensione massima.

Il codificatore contiene anche strati di *auto-attenzione* che si riferiscono all'attenzione all'interno di una stessa sequenza. In questo tipo di strato, tutte le chiavi, i valori e le *query* provengono dall'*output* dello strato precedente, così che ogni posizione nel codificatore possa prestare attenzione a tutte le posizioni di quello strato.

Similmente, gli strati di *auto-attenzione* nel decodificatore permettono ad ogni posizione nel decodificatore di prestare attenzione a tutte le altre posizioni, compresa la posizione stessa. Il termine "auto-attenzione" riflette il fatto che la rete neurale è in grado di attenersi a sé stessa, osservando e ponderando attentamente ogni elemento della

sequenza rispetto agli altri, senza dover fare riferimento a fonti esterne di informazione. Questo consente di catturare informazioni a diversi livelli di astrazione e di fornire una rappresentazione ricca e contestuale dell'*input*.

Come in altri modelli vengono usati degli *embedding* per convertire i token di *input* e *output* in vettori di dimensione d_{modello}.

Inoltre, affinché il modello possa utilizzare l'ordine della sequenza in *input* è necessario inserire le informazioni riguardo alla posizione assoluta e relativa dei token nella sequenza, dal momento che non sono presenti reti convoluzionali o ricorrenti. Per fare ciò vengono aggiunte, prima degli strati del codificatore e del decodificatore, delle codifiche posizionali dette *positional encodings* agli *embedding* di *input* (entrambe di dimensione d_{modello}). Ciò può essere realizzato utilizzando funzioni trigonometriche, funzioni sinusoidali o altri metodi.

Alla fine del processo, vengono usate le usuali funzioni delle reti neurali cioè la funzione lineare e la funzione *softmax* per convertire l'*output* del decodificatore nelle probabilità previste per il token successivo. Oltre alla possibilità di parallelizzare le operazioni e al minor costo di computazione, un ulteriore vantaggio nell'utilizzo di questo tipo di reti rispetto alle più comuni ricorsive e convoluzionali è dato dalla lunghezza dei percorsi tra le posizioni in *input* e *output*. Essi, infatti, sono più brevi negli strati di *auto attenzione* rispetto a quelli nelle reti di altre tipologie. Questo rende più facile per il modello apprendere dipendenze a lungo raggio tra le posizioni nella sequenza.

Grazie alla loro capacità di catturare informazioni contestuali e di compiere operazioni complesse su sequenze di testo, i trasformatori sono diventati la scelta predominante per la creazione di *Large Language Models*. Alcuni esempi noti includono BERT (*Bidirectional Encoder Representations from Transformers*; Devlin *et al.*, 2019), GPT (*Generative Pre-trained Transformer*), T5 (*Text-To-Text Transfer Transformer*), RoBERTa, XLNet, BLOOM ed il più usato GPT, aggiornato nelle varie versioni GPT-2, GPT-3, GPT-3.5 e GPT-4.

1.2.3 Considerazioni sui modelli

Oltre all'innovativo modello di attenzione, un altro importante miglioramento dei LLM è dato dall'uso del pre-addestramento, che rientra nella più generale metodologia dell'apprendimento non supervisionato. Il modello linguistico viene addestrato prima su un *dataset* di grandi dimensioni per poi essere perfezionato su un compito più specifico. Questa tecnica si è dimostrata essere maggiormente efficace nell'eseguire un'ampia gamma di compiti linguistici come la classificazione delle frasi, la risposta alle domande e la

parafrasi (Zhao *et al.*, 2023). I modelli si contraddistinguono anche per la loro abilità di apprendimento *few-shot* che si riferisce alla loro capacità di acquisire informazioni da un numero relativamente ridotto di esempi di addestramento e generalizzare con successo su nuovi compiti o richieste. Questa caratteristica consente loro di fornire prestazioni elevate su numerosi ambiti di elaborazione e di creazione di dati come traduzione automatica, ricerca semantica, sintesi vocale e in generale comprensione del linguaggio naturale.

Con l'aumentare delle risorse di addestramento, dei dati disponibili e delle capacità computazionali, i modelli linguistici continuano a migliorare in termini di precisione, comprensione e capacità di generazione del linguaggio naturale ma presentano ancora delle limitazioni che devono essere affrontate. Servendosi di reti neurali è difficile comprendere a fondo come le previsioni vengano generate dal modello: resta da capire se i modelli stiano imparando a ragionare o semplicemente a memorizzare gli esempi di addestramento in modo più intelligente. Inoltre, i costi economici e computazionali per addestrare e mantenere i *Large Language Models* possono essere molto elevati visto che richiedono infrastrutture di calcolo avanzate e *server* potenti. Alcuni modelli possono riflettere ed amplificare le distorsioni presenti nei dati di addestramento o utilizzare dati privati o personali infrangendo la *privacy* e oltrepassando le misure di sicurezza.

Non si può fare a meno di riflettere sulle questioni etiche che sorgono dall'uso di questi modelli, dal loro impatto, il loro uso e fino a che punto si spingeranno; ma riconoscerne le sfide e come affrontarle è il primo passo per garantire che questi strumenti siano adoperati in modo responsabile e per il beneficio di tutti.

1.3 I rilevatori automatici

La produzione di testi da parte di intelligenze artificiali ha raggiunto livelli sempre più sofisticati, generando sfide e opportunità nel determinare l'autenticità degli scritti. L'emergere di modelli linguistici avanzati, ha reso possibile la generazione automatica di testi che possono essere difficilmente distinguibili da quelli umani. Tuttavia, mentre questa tecnologia offre numerosi vantaggi, contribuisce anche a far sorgere interrogativi etici e pratici sulla sua integrità e affidabilità. Di conseguenza, parallelamente alla nascita e allo sviluppo dei modelli linguistici, sono apparsi in rete i primi rilevatori automatici per testi di questo tipo, ora presenti in discreta quantità sul mercato. Questi strumenti possono essere adoperati in diversi ambiti da persone e organizzazioni che potrebbero trarne beneficio. Il primo utilizzo è comprensibilmente la verifica dell'autenticità dei testi da parte di insegnanti, giornalisti e editori per ridurre al minimo le truffe e gli inganni e potenzialmente la diffusione di notizie false. Le istituzioni accademiche possono inoltre

servirsi dei rilevatori come filtri antiplagio, confrontando i testi con documenti presenti in database di riferimento e in generale per verificare l'originalità del lavoro degli studenti mentre piattaforme e siti *web* hanno l'interesse di analizzare i testi per identificare bot, account spam o profili falsi.

In questo contesto, i classificatori si presentano come uno strumento cruciale per determinare se un testo è stato scritto dall'intelligenza artificiale[2] o da un essere umano. Essi sfruttano una varietà di approcci e tecniche, inclusi modelli linguistici, analisi stilometriche e algoritmi di *machine learning*, per esaminare le caratteristiche intrinseche dei testi e identificare segnali distintivi che possano rivelarne l'origine.

Di recente, studi (Khalil *et al.*, 2023, Wang *et al.*, 2023) apparsi in riviste scientifiche sono stati pubblicati da parte di organizzazioni o di singoli per verificare le abilità di questi strumenti. I risultati però non hanno ripagato il tempo impiegato, non portando ai risultati sperati e mostrando le lacune e l'inefficienza dei vari *software* sottoposti all'indagine. Ciò nonostante, è stato possibile trarre alcune considerazioni analizzando la letteratura presente in rete (Weber-Wulff *et al.*, 2023). Da essa si evince che i testi scritti interamente da umani vengono individuati più facilmente rispetto a quelli prodotti completamente da un LLM per i quali l'accuratezza è raramente superiore al 50%; valori, dunque, non diversi da quanto ottenibile con un classificatore casuale. Spesso i rilevatori faticano a riconoscere un testo, classificandolo interamente come originale, se solo una piccola sezione è stata scritta da un umano servendosi poi di un modello linguistico per generare la parte rimanente. Numerose difficoltà sono state riscontrate anche in testi tradotti, parafrasati o scritti in stili particolari (Pegoraro *et al.*, 2023).

Ci sono inoltre degli aspetti da considerare che riguardano la metodologia adoperata nella maggior parte degli esperimenti, che inevitabilmente influiscono sui risultati (Anderson *et al.*, 2023). Spesso non si ha la certezza della fonte dei testi classificati come generati da umani nei casi in cui siano necessari in grande quantità, e può anche accadere che alcuni di essi siano stati adoperati per l'addestramento di modelli linguistici; il che li renderebbe dei cattivi addestratori (Rogers, 2022). Il numero e la tipologia di metriche utilizzate condizionano il giudizio sull'efficacia del classificatore, rendendo difficile anche la replicabilità del lavoro se queste ultime non fossero esaustivamente descritte. A questo si aggiunge il problema della mancanza di standardizzazione nei criteri di valutazione, che può portare a risultati variabili e difficoltà nella comparazione tra studi diversi. Anche la mancanza di trasparenza riguardo agli algoritmi utilizzati nei *detector* di AI e alle loro potenziali debolezze o *bias* (distorsioni) incorporate è un aspetto da tenere in considerazione.

[2]Nel capitolo si farà riferimento ai LLM come intelligenze artificiali.

1.3.1 Tipologie e caratteristiche

Al momento della scrittura dell'elaborato sono disponibili vari strumenti gratuiti di *AI detection* ma essendo spesso brevettati è difficile cogliere i dettagli tecnici e statistici su cui si basano. Una rassegna dei principali classificatori disponibili e una breve descrizione dei più famosi è comunque doverosa.

Il più noto è probabilmente GPTZero: servizio sviluppato da Edward Tian, studente dell'università di Princeton. Questo strumento controlla la perplessità (*perplexity*[3]) e la discontinuità in un testo permettendogli di fornire in modo efficace una valutazione più accurata della sua origine. Quando GPTZero analizza un documento, non si limita ad analizzare le parole, bensì esamina la struttura delle frasi, il flusso delle idee, l'uso del vocabolario e molto altro. In particolare, utilizza un approccio multilivello articolato in diversi stadi[4].

Come fase preliminare, tramite la regolarità del testo analizza quanto esso sia simile ai modelli di scrittura tipici dell'intelligenza artificiale. Solitamente un documento scritto da un essere umano presenta cambiamenti di stile e di tono durante il suo svolgimento, mentre il contenuto frutto di un'intelligenza artificiale rimane simile per tutta la sua durata. Il livello successivo si basa su un modello addestrato con dati che includono una gran numero di materiale prodotto da studenti, in modo da aumentare l'accuratezza del rilevamento per scopi educativi. Alla base del suo funzionamento vi è *GPTZeroX*, uno strumento di classificazione *sentence to sentence* che analizza ogni frase del testo nel contesto più ampio del documento e determina la probabilità che ogni frase sia stata creata dall'intelligenza artificiale. Un ulteriore livello controlla se il testo in questione è stato trovato in archivi testuali e nella rete accertandone la fonte. Successivamente, utilizzando un LLM dopo ogni parola del testo, il modello sviluppa suggerimenti sui termini successivi e controlla se questi ultimi corrispondono a ciò che è effettivamente presente nel testo. In aggiunta a ciò, il sistema è in grado di difendersi da manovre ed accorgimenti che cercano di sfruttare i punti deboli dei rilevatori AI. GPTZero si serve di un *database* con i metodi più comuni per oltrepassare il rilevamento, come l'omografia e la spaziatura. Come ultimo livello, utilizza un approccio di *deep learning end-to-end*, ovvero una tecnica di apprendimento automatico in cui si adopera una singola rete neurale per compiti complessi utilizzando come *input* direttamente i dati grezzi. La rete viene addestrata sia su massicci corpus di testo provenienti dal *web*, sia su *dataset* generati da una serie di modelli linguistici. Durante il processo vengono usati milioni di documenti

[3]La perplessità misura il grado di casualità del testo nell'elaborazione del linguaggio naturale (Spagnuolo, 2023). Una sua formulazione più tecnica viene fornita nel Capitolo 2.

[4]La struttura dello strumento è stata ricavata dal sito (https://gptzero.me/technology).

che abbracciano vari ambiti della scrittura, i blog, gli articoli di cronaca e altro ancora. I modelli vengono testati su un insieme inedito di articoli umani e artificiali provenienti da un *set* di dati su larga scala, oltre a un insieme più piccolo di articoli difficili da discriminare, che non rientrano nella distribuzione di addestramento. Di questo strumento si servono soprattutto istituzioni didattiche e insegnanti per analizzare testi prodotti da studenti, ovvero la tipologia di dati su cui è stato addestrato. Questa caratteristica non lo rende infallibile ma garantisce una buona accuratezza dei risultati (Cui, 2023).

Analogamente al precedente, Open AI classifier analizza le caratteristiche linguistiche e stilistiche di un testo per assegnargli la probabilità che sia stato prodotto in modo artificiale. Sviluppato dagli stessi creatori di ChatGPT, il classificatore è un modello linguistico messo a punto su un *dataset* di coppie di testi scritti da esseri umani e testi scritti dall'intelligenza artificiale sullo stesso argomento. Il *set* di dati di addestramento si basa su una varietà di fonti ritenute scritte da umani come dati di *pre-training* e le risposte umane sui *prompt* inviati a *InstructGPT* (una tipologia di LLM). In questa fase, ogni testo è stato diviso in una richiesta e in una risposta. Sulle richieste sono state generate risposte da una serie di modelli linguistici addestrati in modo differente. Come prodotto finale il modello è stato regolato in modo da avere una soglia di confidenza che mantenga basso il tasso di Falsi Positivi in modo da massimizzare la *Precision*[5]; in altre parole, il testo viene contraddistinto come probabilmente scritto dall'intelligenza artificiale solo se il classificatore è molto fiducioso.

Come riportato dai produttori, il classificatore possiede però alcune limitazioni: se il testo viene modificato anche di poco la classificazione può essere agevolmente elusa e, in aggiunta, il classificatore può sbagliare facilmente se deve analizzare del testo prodotto da bambini e non in lingua inglese. Questo perché è stato addestrato sottoponendogli testi prodotti da adulti e scritti in quel determinato idioma. In generale, i classificatori basati sulle reti neurali sono poco calibrati al di fuori dei dati di addestramento: per *input* molto diversi dal testo su cui è stato addestrato, il classificatore è talvolta estremamente sicuro di una previsione errata. Di conseguenza non deve essere utilizzato come strumento decisionale principale, ma come complemento ad altri metodi per determinare la fonte di un testo.

Per questi motivi, come affermato nel sito di Open AI e in numerosi articoli (Chaka e Chaka, 2023) il classificatore non si è rivelato sufficientemente affidabile; infatti, in un test di prova è stato capace di identificare solamente il 26% dei testi scritti dall'AI, mentre ha erroneamente etichettato un testo scritto dall'uomo come prodotto dall'intelligenza

[5]Veri Positivi/(Veri Positivi + Falsi Positivi).

artificiale il 9% delle volte (Falsi Positivi). A causa di questi limiti Open AI classifier è stato disabilitato nel mese di luglio del 2023 e reso non più disponibile (Goodwin, 2023).

Il classificatore ZeroGPT si basa su complessi algoritmi prodotti dal gruppo di ricerca omonimo dopo aver analizzato più di dieci milioni di articoli e testi prodotti da umani e da LLM in varie lingue. Lo strumento di rilevamento utilizza la tecnologia *DeepAnalyse*™ per identificare l'origine del testo e fornisce risultati con un tasso di accuratezza fino al 98%. Similmente ai classificatori citati in precedenza, ZeroGPT esamina gli schemi che spesso caratterizzano i contenuti generati dall'intelligenza artificiale, come alcune frasi ripetitive o l'eccessivo ricorso a frasi strutturate e formulate. Inoltre, considera la profondità della comprensione del contesto e la presenza di elementi tra loro poco compatibili, che possono contraddistinguere la scrittura umana.

Altri rilevatori noti sono: Contentatscale ai detector, Writer, GPT-2 Output Detector Demo, AI Content Detector, Crossplag detector, QuillBot, Plagirism detector, Kazanseo detector, Turnitin e Compilatio.

1.3.2 Analisi delle capacità

Tra l'elenco proposto sono stati selezionati dieci classificatori da testare al fine di valutare le loro *performance* di riconoscimento. La scelta è il risultato di una ampia ricerca *online*, in base a quanto fruibile al momento della scrittura dell'elaborato. Va sottolineato che il panorama è in continuo sviluppo ed evoluzione e alcuni siti saranno già obsoleti quando la stesura del lavoro sarà terminata, mentre altri saranno appena stati lanciati sul mercato. La seguente analisi non vuole essere esaustiva e completa su tutto il repertorio disponibile, bensì dare un'idea delle prestazioni medie dei *detector* e della loro affidabilità. In seguito sono elencati i rilevatori scelti e il loro sito di riferimento:

- GPTZero: https://gptzero.me/
- ZeroGPT: https://www.zerogpt.com/
- Contentatscale ai detector: https://contentatscale.ai/ai-content-detector/
- Writer: https://writer.com/ai-content-detector/
- GPT-2 Output Detector Demo: https://openai-openai-detector.hf.space/
- AI Content Detector: https://corrector.app/ai-content-detector/
- Crossplag detector: https://app.crossplag.com/individual/detector
- Plagiarism detector: https://plagiarismdetector.net/it/ai-content-detector
- Kazanseo detector: https://kazanseo.com/detector
- QuillBot: https://quillbot.com/ai-content-detector

Questi strumenti presentano caratteristiche, requisiti e metodi di esposizione dei risultati differenti come descritto nella Tabella 1.1.

Tabella 1.1: Rilevatori di testo e relative caratteristiche: numero minimo e massimo di token in *input* e formato dell'*output* generato.

Rilevatore	**Dim. minima**	**Dim. massima**	**Output**
GPTZero	250 caratteri	5000 caratteri	% human, mixed, AI generated
ZeroGPT	600 caratteri	15000 caratteri	% AI generated
Contentatscale ai detector	Non Specificato	2500 caratteri	Human/AI
Writer	Non specificato	1500 caratteri	% Human generated
GPT-2 Output Detector Demo	50 token	510 token	% Human/AI generated
AI Content Detector	Non specificato	800 token	% fake
Crossplag detector	Non specificato	3000 token	% AI generated
Plagiarism detector	Non specificato	2000 token	% AI generated
Kazanseo detector	Non specificato	2000 token	% AI generated
QuillBot	Non specificato	1200 token	% AI detected

Per l'analisi è stato utilizzato un corpus, creato da Rexhep Shijaku e Ercan Canhasi (2023) da cui sono stati selezionati 30 testi scritti da umani (U) e 30 scritti da ChatGPT[6] (AI), entrambi in lingua inglese per garantire il pieno funzionamento dei rilevatori. I testi appartenenti alla prima categoria sono stati estratti automaticamente da un *file* PDF[7] contenente una collezione di temi per esami TOEFL, per cui si ha un'alta probabilità che non siano stati usati per l'addestramento di LLM. Inoltre, l'esame TOEFL viene sostenuto da persone non madrelingua inglesi e quindi i testi creati dovrebbero mettere maggiormente in difficoltà i classificatori. Invece, i testi generati artificialmente sono stati ricavati dall'API di ChatGPT utilizzando un *client*[8], chiedendo all'intelligenza artificiale di scrivere dei temi con le stesse consegne sottoposte agli studenti dell'esame TOEFL. È possibile leggere le domande rivolte al *chatbot*, in questo *file*[9]. La dimensione dei testi, tale da consentire a tutti i rilevatori di svolgere l'analisi, è stata limitata a un massimo di 500 token.

È importante sottolineare anche che la natura dei testi (non parafrasati, tradotti o alterati in altro modo) non permette di valutare le complete capacità degli strumenti selezionati, ma solo di verificare la loro affidabilità in uno dei possibili campi di utilizzo, con una tipologia di testi che usualmente potrebbe essere sottoposta a modifiche o ritocchi.

Ai fini dell'analisi, dal momento che i diversi rilevatori mostrano i loro risultati in modo diverso, è stato necessario standardizzare le risposte attraverso una forma comune

[6] In questo caso è stata utilizzata la versione GPT-3.5.
[7] https://englishclubmskh.files.wordpress.com/2018/09/toefl-essays.pdf.
[8] https://github.com/acheong08/ChatGPT.
[9] https://github.com/rexshijaku/chatgpt-generated-text-detection-corpus/blob/main/full_texts/questions.txt.

che si basa sulla percentuale di testo attribuita all'intelligenza artificiale. Nei casi in cui il rilevatore mostrasse solo se il testo fosse attribuibile maggiormente ad un umano o ad una AI è stata attribuita una percentuale pari a 100% a quella categoria. Le Tabelle 1.4 e 1.5 riportano i risultati della rilevazione, rispettivamente per i testi della categoria AI e della categoria U. Coi dati raccolti attraverso la somministrazione del corpus ai classificatori, i testi sono stati ripartiti tramite le percentuali di risposta nel modo seguente:

- Testi con valori minori o uguali a 25% sono stati classificati come generati da umani (UG)
- Testi con valori maggiori di 25% ma minori o uguali a 50% sono stati classificati come probabilmente generati da umani (PUG)
- Testi con valori maggiori di 50% ma minori o uguali a 75% sono stati classificati come probabilmente generati dall' intelligenza artificiale (PAIG)
- Testi con valori maggiori di 75% sono stati classificati come generati dall'intelligenza artificiale (AIG)

Questa suddivisione, presentata tramite la Tabella 1.2, permette di esaminare i risultati sulla base di metriche note in letteratura, in modo da poter valutare efficacemente le capacità di rilevatori.

Tabella 1.2: Conteggi risultanti dalla classificazione dei testi, in base alla loro tipologia e agli *output* ricodificati dei rilevatori automatici.

Tipologia testo	**Umano**				**Intelligenza Artificiale**			
Risultato rilevatore	**UG**	**PUG**	**PAIG**	**AIG**	**UG**	**PUG**	**PAIG**	**AIG**
GPTZero	29	1	0	0	0	1	4	25
ZeroGPT	28	2	0	0	0	0	2	28
Contentatscale ai detector	29	0	0	1	1	0	0	29
Writer	30	0	0	0	1	29	0	0
GPT-2 Output Detector Demo	27	0	0	3	0	0	1	29
AI Content Detector	24	3	3	0	0	0	1	29
Crossplag detector	26	1	2	1	0	0	3	27
Plagiarism detector	27	0	1	2	0	0	1	29
Kazanseo detector	28	1	1	0	0	0	2	28
QuillBot	28	1	1	0	0	0	1	29

Considerando che complessivamente i rilevatori hanno prodotto buoni risultati, per cogliere al meglio le capacità dei singoli prodotti la loro abilità è stata valutata non solo tramite gli usuali conteggi di Veri Positivi (TP), Falsi Positivi (FP), Veri Negativi (TN) e Falsi Negativi (FN), bensì sulle seguenti quantità:

- *Veri Positivi* (TP): testi della categoria AI classificati come AIG
- *Falsi Positivi* (FP): testi della categoria U classificati come AIG
- *Veri Negativi* (TN): testi della categoria U classificati come UG
- *Falsi Negativi* (FN): testi della categoria AI classificati come UG
- *Incerti Positivi* (IP): testi delle categorie U o AI classificati come PAIG
- *Incerti Negativi* (IN): testi delle categorie U o AI classificati come PUG
- *Incerti* (I): somma di *Incerti Positivi* (IP) e *Incerti Negativi* (IN)

Questa scelta, di fatto, si traduce nell'utilizzo di una soglia pari al 75% per la discriminazione delle due categorie di testi, ed è motivata dalla necessità di avere un buon livello di sicurezza nella decisione di attribuire un testo a una tipologia o all'altra. Le applicazioni pratiche dei rilevatori impongono che non ci si possa permettere di sbagliare se devono essere prese decisioni sulla provenienza di un testo; quindi, per ridurre al minimo gli errori la soglia di discriminazione deve essere alta.

Attraverso le quantità descritte, posto N = 60 pari alla numerosità del corpus, i classificatori sono stati valutati secondo i seguenti indicatori:

- *Errore Totale*: (FN+FP+IP+IN)/N=(FN+FP+I)/(TP+TN+FP+FN+I)
- *Accuratezza*: (TP+TN)/(TP+TN+FP+FN+IP+IN)=(TP+TN)/N
- *Precision*: TP/(FP+TP)
- *Recall* (*Sensitivity*): TP/(N/2)
- *Specificity*: TN/(N/2)
- *F1*: $2/(1/Recall + 1/Precision)$

Queste metriche, in questo caso appositamente modificate, sono spesso usate in ambito statistico per valutare le capacità di un test di classificazione binaria. L'*Errore Totale* di classificazione indica il tasso di errata classificazione ed è il complemento ad 1 dell'*Accuratezza*, cioè il tasso di corretta classificazione. Queste due quantità pesano gli errori allo stesso modo. La *Precision* è calcolata come il rapporto tra *Veri Positivi* e i testi classificati come positivi ed è chiamata anche *Positive Predicted Value*. *Sensitivity* e *Specificity* sono metriche molto usate in ambito biostatistico (Gaddis *et al.*, 1990); la prima è chiamata anche *True Positive Rate* ovvero la proporzione di casi positivi (quindi temi AI) che sono correttamente identificati, mentre la seconda è detta *True Negative Rate* ed indica la proporzione di casi negativi (quindi testi U) correttamente identificati. L'indicatore *F1* è calcolato come una media armonica tra *Precision* e *Recall*, varia tra 0 e 1 ed un valore più alto della statistica indica una classificazione migliore.

Tabella 1.3: Metriche di valutazione e rispettivi valori per i rilevatori automatici selezionati.

Rilevatore	Errore Totale	Accuratezza	Precision	Recall	Specificity	F1
GPTZero	0.100	0.900	1	0.833	0.967	0.909
ZeroGPT	0.067	0.933	1	0.933	0.933	0.966
Contentatscale ai detector	0.033	0.967	0.966	0.967	0.967	0.967
Writer	0.500	0.500	NaN	0	1	NaN
GPT-2 Output Detector Demo	0.060	0.933	0.906	0.967	0.900	0.935
AI Content Detector	0.117	0.883	1	0.967	0.800	0.983
Crossplag detector	0.117	0.883	0.964	0.900	0.867	0.931
Plagiarism detector	0.067	0.933	0.935	0.967	0.900	0.951
Kazanseo detector	0.067	0.933	1	0.933	0.933	0.966
QuillBot	0.050	0.950	1	0.967	0.933	0.983

Analizzando le Tabelle 1.2 e 1.3 si può avere una visione di insieme sulle capacità discriminatorie dei rilevatori. Nel complesso tutti i rilevatori hanno ottenuto buoni risultati e, se ci si basasse su una discriminazione con soglia al 50% tra testi UG e AIG, sarebbero quasi perfetti. Come detto in precedenza, per calcolare metriche come *Accuratezza* ed *Errore Totale* di classificazione si è invece deciso di usare un *threshold*, per la probabilità che un testo sia generato artificialmente, superiore al 75%. Nonostante la soglia elevata quasi tutti gli *Errori Totali* presentano valori intorno a 0.1, mentre per l'*Accuratezza* si collocano attorno a 0.9. Questo non accade per il rilevatore Writer che ha identificato correttamente tutti i testi generati da umani ed erroneamente tutti i testi generati da ChatGPT. Osservando le singole probabilità si può notare come per i testi AI esse non superino mai il 50% ma siano quasi sempre prossime a 0 per i testi U. Probabilmente questo significa che il rivelatore è in grado di identificare correttamente l'origine dei testi ma che la probabilità in *output* è distorta e che andrebbe ribilanciata per portarla in un *range* tra 0% e 50%, in modo da avere valori plausibili. Escludendo questo rilevatore, si può notare come i valori di *Precision* siano molto spesso pari ad 1, denotando l'abilità dei rilevatori nell'individuare i testi AI. Questo dato è confermato dagli alti valori di *Sensitivity* (*Recall*). Infine, i buoni valori di *Specificity* mostrano come i rilevatori siano anche in grado di individuare correttamente i testi U. Infine, i classificatori che hanno portato risultati migliori, secondo la metrica *F1*, sono AI Content Detector e QuillBot.

Da riflessioni di natura più teorica riguardo l'affidabilità di questi strumenti, basandosi sul loro utilizzo è emerso che:

- A volte, ripetendo nuovamente un test per un certo testo, il rilevatore ha fornito risultati diversi, dimostrando quindi poca sicurezza nelle risposte, implicando la

non completa replicabilità dei risultati e compromettendo, di fatto, le analisi
- I rilevatori presentano dei *bias* nelle risposte, dipendenti probabilmente dal loro metodo di costruzione, che andrebbero conosciuti prima del loro utilizzo
- Anche i rilevatori migliori hanno mostrato errori di classificazione che si sono verificati con una frequenza pari ad uno ogni dieci o quindici testi. In questo caso sarebbe conveniente ripetere l'analisi per lo specifico documento o utilizzare un altro rilevatore per conferma
- Non sembra che nel *dataset* preso in analisi fossero presenti testi difficili da rilevare, ovvero classificati in modo errato da più strumenti, ma che gli errori che si sono verificati fossero più o meno casuali e dovuti a difetti dei singoli rilevatori
- Alcuni rilevatori come Contentatscale ai detector o GPT-2 Output Detector Demo hanno mostrato, in *output*, probabilità sempre prossime a zero o a cento, non dando la possibilità di identificare che porzione del testo fosse effettivamente rilevata come prodotta da umano o meno. Probabilmente la classificazione in AIG o UG viene eseguita internamente dal rilevatore con una soglia predefinita
- Per questo tipo di dati i rilevatori si sono dimostrati in ugual modo capaci di identificare testi AI o U

Dalle considerazioni precedenti si può dunque affermare che questi strumenti di rilevazione non siano particolarmente affidabili, ma che per testi su cui non sono state applicate modifiche, parafrasi o traduzioni funzionino discretamente bene. Valori di *Precision* pari a 0.95 indicano che su venti testi classificati come AIG solamente uno sarebbe un FP. Ipotizzando un utilizzo in ambito scolastico, questo vorrebbe dire che in una classe di venti alunni dove quasi tutti hanno probabilmente copiato, uno studente sarebbe ingiustamente incolpato di non aver svolto correttamente il suo lavoro. Dal momento che questi strumenti non forniscono risposte certe, la probabilità che attualmente possano essere usati in un contesto accademico è piuttosto bassa. I rilevatori forniscono un'interpretazione probabilistica del testo che esaminano, ma anche un'evidenza del 95% di generazione artificiale non significherebbe molto dal momento che non c'è un documento di origine che possa dimostrare l'esistenza di un plagio. Non ci si può basare solo su di essi per determinare la provenienza di un elaborato; tuttavia, se uno strumento dovesse mostrare evidenza di parti di testo generate da un computer, si potrebbe provare l'utilizzo di diversi strumenti per dare robustezza ai risultati.

L'analisi dei rilevatori di testo generato dall'intelligenza artificiale ha rivelato una serie di sfide e limitazioni significative. Sebbene tali strumenti possano essere utili per identificare testi evidentemente non scritti da umani, la loro affidabilità è compromessa

dalla variabilità dei risultati, dalla presenza di distorsioni e dagli errori di classificazione. L'incapacità dei rilevatori di fornire risposte certe e la presenza di probabilità estreme rendono difficile il loro utilizzo in contesti (accademici, scientifici, politici, ...) dove la precisione è fondamentale. Attualmente non esiste una scorciatoia o una strada più veloce per questo tipo di problemi; tuttavia, l'utilizzo combinato di diversi rilevatori potrebbe aumentare la robustezza dei risultati, sebbene non sostituisca un'analisi umana accurata. In definitiva, mentre i rilevatori di testo generato dall'AI offrono un potenziale significativo, è importante considerarne criticamente le limitazioni e procedere con cautela nel loro utilizzo.

Tabella 1.4: Risultati della rilevazione per i testi AI tramite i dieci rilevatori. Viene mostrata la percentuale di testo attribuito all'AI.

Testo	GPTZero	ZeroGPT	Contentatscale ai detector	Writer	GPT-2 Output Detector Demo	AI Content Detector	Crossplag detector	Plagiarism detector	Kazanseo detector	QuillBot
AI-2	95%	100%	100%	31%	99.98%	100%	100%	100%	99.98%	100%
AI-3	42%	100%	100%	27%	99.98%	100%	100%	100%	88.84%	100%
AI-5	77%	100%	100%	26%	99.98%	100%	100%	100%	99.50%	100%
AI-6	97%	100%	100%	32%	99.98%	100%	83%	100%	99.98%	100%
AI-7	97%	100%	100%	41%	99.98%	100%	100%	100%	99.98%	100%
AI-8	96%	100%	100%	37%	99.98%	100%	100%	100%	99.98%	100%
AI-9	95%	100%	100%	29%	99.98%	100%	100%	100%	99.98%	100%
AI-11	97%	100%	100%	31%	99.98%	100%	100%	100%	99.30%	97%
AI-12	97%	100%	100%	42%	99.98%	100%	72%	100%	94.30%	100%
AI-14	74%	100%	100%	28%	99.98%	100%	100%	100%	99.98%	100%
AI-15	88%	100%	100%	36%	99.98%	100%	100%	100%	99.98%	100%
AI-16	95%	100%	100%	31%	99.98%	100%	100%	100%	82.66%	100%
AI-17	94%	100%	100%	27%	99.98%	100%	100%	100%	96.67%	100%
AI-18	98%	73%	100%	43%	99.98%	100%	100%	100%	99.98%	100%
AI-19	96%	100%	100%	45%	99.98%	70%	100%	100%	99.98%	100%
AI-20	96%	100%	100%	21%	99.98%	100%	69%	100%	99.98%	100%
AI-21	55%	100%	100%	29%	99.98%	100%	100%	100%	94.56%	100%
AI-22	97%	100%	100%	36%	65.67%	100%	100%	100%	99.98%	70%
AI-23	79%	100%	100%	35%	99.98%	100%	100%	80%	67.50%	100%
AI-24	98%	100%	0%	40%	99.98%	100%	100%	100%	70.30%	100%
AI-25	93%	100%	100%	37%	99.98%	100%	100%	100%	99.98%	100%
AI-26	92%	100%	100%	38%	99.98%	100%	100%	100%	99.98%	100%
AI-27	97%	100%	100%	46%	99.98%	100%	100%	70%	99.98%	100%
AI-28	96%	100%	100%	28%	99.98%	100%	100%	100%	99.98%	100%
AI-35	63%	100%	100%	31%	99.98%	100%	100%	100%	99.98%	100%
AI-36	90%	67%	100%	34%	99.98%	100%	100%	100%	97.98%	100%
AI-37	94%	100%	100%	35%	99.98%	100%	73%	100%	96.50%	100%
AI-40	65%	100%	100%	46%	99.98%	100%	100%	100%	99.98%	100%
AI-41	98%	100%	100%	29%	99.98%	100%	100%	100%	94.50%	100%
AI-42	97%	100%	100%	26%	99.98%	100%	100%	100%	99.98%	100%

Tabella 1.5: Risultati della rilevazione per i testi U tramite i dieci rilevatori. Viene mostrata la percentuale di testo attribuito all'AI.

Testo	GPTZero	ZeroGPT	Contentatscale ai detector	Writer	GPT-2 Output Detector Demo	AI Content Detector	Crossplag detector	Plagiarism detector	Kazanseo detector	QuillBot
U-2	0%	0%	0%	3%	0.02%	0%	9%	100%	18.89%	0%
U-3	0%	0%	0%	2%	99.96%	0%	0%	0%	1.28%	11%
U-5	0%	0%	0%	0%	0.04%	0%	0%	0%	0.04%	16%
U-6	0%	0%	0%	3%	0.03%	0%	100%	0%	9.55%	25%
U-7	0%	0%	0%	0%	0.02%	0%	0%	0%	0.02%	14%
U-8	0%	12.03%	0%	0%	0.02%	0%	0%	0%	0.06%	9%
U-9	0%	0%	0%	0%	0.03%	12.45%	0%	0%	10.12%	0%
U-11	0%	6.00%	0%	1%	0.04%	0%	0%	0%	9.65%	12%
U-12	0%	0%	0%	5%	0.01%	0%	0%	0%	0.08%	13%
U-14	0%	0%	0%	4%	0.02%	13.56%	0%	20%	0.07%	2%
U-15	0%	26.16%	0%	0%	0.02%	0%	0%	0%	8.34%	21%
U-16	0%	0%	0%	0%	0.02%	0%	67%	0%	7.56%	0%
U-17	0%	0%	0%	0%	0.04%	51.34%	0%	0%	0.04%	34%
U-18	0%	0%	0%	0%	98.98%	0%	0%	0%	45.43%	9%
U-19	0%	4.00%	100%	6%	0.03%	26.47%	0%	0%	0.02%	12%
U-20	12%	0%	0%	2%	0.02%	61.26%	0%	0%	0.03%	11%
U-21	0%	0%	0%	2%	0.03%	14.84%	12%	0%	0.05%	51%
U-22	0%	19.32%	0%	12%	0.02%	18.69%	34%	0%	6.04%	8%
U-23	31%	0%	0%	0%	0.02%	0%	0%	0%	5.04%	7%
U-24	0%	6.40%	0%	0%	0.05%	0%	0%	51%	12.34%	6%
U-25	0%	20.01%	0%	0%	0.04%	0%	0%	0%	0.05%	8%
U-26	0%	0%	0%	0%	0.02%	42.34%	0%	0%	0.04%	12%
U-27	0%	20.28%	0%	0%	0.02%	12.00%	0%	0%	51.12%	0%
U-28	0%	0%	0%	5%	0.02%	74.45%	0%	0%	0.03%	0%
U-35	0%	0%	0%	5%	97.98%	0%	0%	0%	0.07%	0%
U-36	0%	27.56%	0%	0%	0.03%	0%	74%	0%	1.24%	25%
U-37	0%	0%	0%	0%	0.04%	0%	0%	0%	0.06%	0%
U-40	0%	0%	0%	0%	0.02%	34.56%	0%	0%	0.05%	25%
U-41	0%	0%	0%	1%	0.02%	6.00%	0%	90%	0.02%	0%
U-42	0%	0%	0%	0%	0.02%	0%	0%	0%	0.03%	17%

Capitolo 2

Descrizione dei dati e dei metodi

2.1 Modelli linguistici di interesse

2.1.1 GPT-3

Generative Pre-trained Transformer 3 (GPT-3) è un LLM autoregressivo reso disponibile dal laboratorio di ricerca Open AI a fine maggio del 2020. È il modello di terza generazione della serie GPT, e come per il predecessore GPT-2 la sua architettura si basa su un modello di trasformatore con 175 miliardi di parametri, che richiedono 800 GB di memoria per l'esecuzione (Yousri *et al.*, 2023).

I dati di allenamento, di cui si stima (Liu *et al.*, 2024) un peso pari a 700GB (o circa 500 miliardi di token), provengono principalmente da Common Crawl (410 miliardi di token) e da WebText2 (19 miliardi di token). Altre fonti utilizzate sono: la versione inglese di *Wikipedia* (3 miliardi di token) e una vasta quantità di testi contenuti nei corpus *Books1* (12 miliardi di token) e *Books2* (55 miliardi di token).

Common Crawl è un'organizzazione senza scopo di lucro che dal 2011 esegue la scansione della rete e fornisce gratuitamente i propri archivi al pubblico. Le sue informazioni sono ottenute attraverso la tecnica dello *scraping* e sono messe a disposizione per le aziende che si occupano di sviluppare algoritmi AI (Schäfer e Roland, 2016). Analogamente, il *dataset* WebText2 è un vasto corpus di testi in lingua inglese raccolti da pagine *web* pubbliche disponibili su Internet.

Dal momento che alcune fonti sono note per avere una qualità superiore, i dati provenienti da *Wikipedia* sono stati campionati numerose volte mentre altri come Common Crawl sono stati utilizzati in minor quantità (Lambda Labs, 2024). In questo modo viene data priorità ai dati migliori aumentando la qualità complessiva del corpus.

GPT-3 è in grado di eseguire l'apprendimento "*zero-shot*", "*few-shot*" e "*one-shot*", ovvero diversi scenari di apprendimento automatico che si differenziano per il numero di esempi di addestramento forniti al modello per un compito specifico (Brown *et al*,. 2020). Nel primo caso il modello non riceve esempi diretti, nel secondo ne riceve pochi e nel terzo particolare caso riceve solo un singolo esempio di addestramento per il compito di interesse.

Nella prima fase di addestramento, il *pre-training*, ovvero il processo di analisi e comprensione dei *pattern*, aiuta il modello a sviluppare una comprensione generale del linguaggio e della sua struttura (Radford e Narasimhan, 2018). Una volta che questa fase è completata, GPT-3 può essere perfezionato per alcuni compiti specifici, come rispondere alle domande, tradurre o generare del testo. Il processo di *fine-tuning* (messa a punto o perfezionamento) coinvolge l'addestramento di GPT-3 su piccoli e più specifici *dataset* al fine di eseguire le sue funzioni in modo ancora migliore su quegli argomenti (Lin *et al*,. 2024). In particolare, l'addestramento si è svolto usando una istanza *cloud* della GPU Tesla V100 ed il suo costo è stato stimato (Lambda Labs, 2024) essere pari a 4.6 milioni di dollari.

Nello specifico GPT-3 genera il suo *output* un token alla volta, quindi parola per parola. La fase computazionale e di calcolo avviene in una pila di 96 strati che contribuiscono a formare la struttura del trasformatore. Ognuno di questi strati possiede 1.8 miliardi di parametri che permettono di quantificare le probabilità per i singoli token. La dimensione del contesto di GPT-3 è pari a 2048 token. Il contesto si riferisce alla quantità di testo o informazioni che il modello considera durante la generazione di *output* per una determinata sequenza di *input*. In questo caso, il *contest window* di GPT-3 è di 2048 token, per cui, il modello considera fino a 2048 token precedenti quando genera un token successivo nella sequenza.

L'architettura del trasformatore su cui si basa GPT-3 prevede l'alternanza tra strati di *auto-attenzione* densi e sparsi. Nella seconda tipologia, ciascun token è collegato solo a un sottoinsieme limitato di altri token, anziché a tutti quelli presenti nella sequenza come avviene in uno strato denso. Questa combinazione è stata progettata per bilanciare la capacità del modello di catturare relazioni complesse a lungo raggio tra le parole mantenendo comunque una computazione efficiente (Vaswani *et al.*, 2017).

La serie GPT-3 si basa sulla sua prima versione nominata *davinci* che ha permesso, in seguito, di creare il modello di InstructGPT chiamato *Text-davinci-00*; ovvero una sua variante addestrata con un'attenzione specifica sulla capacità di comprendere e rispondere a istruzioni umane. La serie di modelli GPT-3.5 è una versione più potente del suo predecessore GPT-3, e si basa sul modello *code-davinci-002* (Ye *et al.*, 2023). Successi-

vamente, Open AI ha perfezionato quest'ultimo sviluppando *text-davinci-002* e introducendo la strategia di addestramento RLHF, utilizzata per creare *text-davinci-003* che ha migliorato ulteriormente l'abilità di comprendere istruzioni e generare testo. Basandosi su quest'ultima versione, Open AI ha ottimizzato GPT 3.5 turbo che è attualmente il miglior modello della serie GPT-3 disponibile.

2.1.2 ChatGPT

ChatGPT (acronimo di *Chat Generative Pre-trained Transformer*) è un modello linguistico della serie GPT-3.5 addestrato per interagire in modo interattivo e si può considerare come il risultato di *fine-tuning* di un modello GPT-3. ChatGPT deriva dall'utilizzo e dall'addestramento di InstructGPT come *code-davinci-002*, *text-davinci-002* e *text-davinci-003*.

Il modello è stato addestrato adoperando il *Reinforcement Learning from Human Feedback* (RLHF): una strategia utilizzata soprattutto nell'ambito dell'intelligenza artificiale per migliorare le prestazioni dei modelli attraverso il *feedback* umano (Dai *et al.*, 2023). In questo approccio di apprendimento, il modello interagisce con un ambiente simulato o del mondo reale e riceve un riscontro positivo o negativo basato sulle sue azioni. Gli esseri umani indicano se le azioni del modello sono corrette o incorrette, utili o non utili, coerenti o non coerenti e così via; questi segnali possono essere considerati come delle ricompense o penalità. Il *feedback* viene poi utilizzato per aggiornare il modello e ottimizzarlo, in particolare usando l'algoritmo di *Proximal Policy Optimization* (PPO) per migliorare la sua politica decisionale (Schulman *et al.*, 2017). Nel caso di ChatGPT, durante l'apprendimento sono state utilizzate conversazioni nelle quali gli istruttori interpretavano sia la parte dell'utente, sia la parte dell'assistente. Nella successiva fase di rinforzo, gli istruttori umani hanno prima valutato o ordinato le risposte che il modello aveva dato nella conversazione precedente per poi creare i modelli di ricompensa su cui è avvenuto il perfezionamento.

Data la sua natura, ChatGPT è definibile come un *chatbot*: un *software* progettato per simulare una conversazione con un essere umano. In breve tempo ha raggiunto capacità e prestazioni sorprendenti, tanto che, dalla data del suo rilascio (30 novembre 2022) a febbraio 2023, ha visto crescere la sua popolarità in tutto il mondo, raggiungendo in soli due mesi i 100 milioni di utenti. Come gli altri modelli è in grado di completare e suggerire del testo, tradurre, analizzare e riassumere testi, creare contenuti e molto altro. Tuttavia, data la natura dei dati di addestramento, la conoscenza di ChatGPT è molto limitata per informazioni ed eventi successivi al 2021.

2.2 Creazione del corpus e descrizione dei dati

2.2.1 Creazione del corpus

Il principale scopo di questo lavoro è la costruzione e il confronto di metodi e modelli di classificazione per rilevare automaticamente la provenienza di un testo, a seconda che l'autore sia un umano o un modello linguistico. Per fare ciò, il primo passo è creare il corpus su cui verteranno le analisi. È necessario dunque disporre di una discreta quantità di testi, da fonti certe, da poter esaminare. Per avere un'idea generale del fenomeno di interesse e per non complicare troppo la procedura, è stato scelto di ricavare testi scritti completamente da umani o da modelli linguistici, in lingua italiana, senza parafrasarli, ritoccarli o modificarli in alcun modo. Per i testi del primo tipo è stato chiesto a 300 studenti, provenienti da alcune università italiane, di scrivere un breve scritto riguardante un argomento a scelta tra un elenco di temi disponibili, relativi principalmente ad aspetti di vita quotidiana. La consegna prevedeva che i testi avessero una lunghezza compresa tra le 400 e le 440 parole; una dimensione sufficiente da consentire analisi sulla loro forma e struttura, ma non eccessiva da impedire l'impiego di tecniche che implicano un numero massimo di token per il loro utilizzo[1]. Adoperare testi prodotti da studenti è coerente con la natura delle analisi precedenti sui rilevatori automatici e con l'impiego in ambito accademico che possono avere i risultati di questa tesi.

Successivamente, è stato prodotto il corpus di testi generati da un modello linguistico. Si è deciso di utilizzare ChatGPT nella sua versione 3.5 per via del suo facile e intuitivo utilizzo e in quanto, al momento di scrittura di questo lavoro, è il modello più diffuso tra gli studenti. Servendosi degli stessi titoli dei temi adoperati in precedenza, ogni studente ha sottoposto la seguente richiesta al modello linguistico: "Ti chiedo di scrivere un tema con lo stile di uno studente universitario italiano di 20 anni, la lunghezza del tema deve essere di almeno 400 parole (minimo 400, massimo 440 parole), l'argomento del tema è il seguente:". Per evitare di essere influenzati gli studenti hanno prima scritto il loro tema e poi sottoposto la richiesta a ChatGPT. Nel caso il testo prodotto non fosse della dimensione richiesta, è stato chiesto di ampliarlo o ridurlo fino al raggiungimento dell'obiettivo.

Il corpus di analisi è dunque composto da 600 testi in lingua italiana, 300 etichettati come "Human" e 300 come "AI" di lunghezze comparabili e di simile natura. In particolare, la rilevazione per l'insieme di testi, realizzata principalmente dal Professor M. A.

[1] Alcune tecniche spiegate in seguito, come il modello BERT, prevedono un testo in *input* di massimo 512 token.

Cortelazzo, si è svolta in più fasi e in sedi differenti, e ha condotto ad un insieme di otto subcorpus che costituiscono il corpus complessivo così ripartito:

- 32 testi rilevati dagli studenti del primo anno (a.a. 2023-24) della laurea triennale in Scienze della mediazione linguistica del Campus Ciels, sede di Bologna
- 108 testi rilevati dagli studenti del primo anno (a.a. 2023-24) della laurea triennale in Scienze della mediazione linguistica del Campus Ciels, sede di Brescia
- 330 testi rilevati dagli studenti principalmente del primo anno (a.a. 2023-24) della laurea triennale in Comunicazione dell'Università di Padova
- 28 testi rilevati dagli studenti della laurea magistrale in Strategie di comunicazione (a.a. 2021-22, 2022-23, 2023-24) dell'Università di Padova
- 50 testi rilevati dagli studenti della laurea magistrale in Scienze della formazione primaria (a.a. 2021-22, 2022-23, 2023-24) dell'Università di Padova
- 30 testi rilevati dagli studenti della laurea triennale in Lettere (a.a. 2019-20, 2021-22, 2021-23) dell'Università di Padova e della laurea in Lettere antiche e moderne, arti, comunicazione dell'Università di Trieste (a.a. 2021-23)
- 22 testi rilevati dagli studenti della laurea magistrale in Scienze Statistiche (a.a. 2023-24) dell'Università di Padova

I testi sono stati scritti o generati da ChatGPT nel periodo gennaio-maggio del 2024.

2.2.2 Lexical Processing

Il *lexical processing* del corpus è una fase fondamentale nel processo di classificazione dei testi perché aiuta ad estrarre le caratteristiche rilevanti dai dati grezzi e li rende pronti per l'analisi. Questa procedura comprende una varietà di tecniche tra cui: tokenizzazione, *lowercasing*, *stemming*, rimozione di parole ed altre. Questa fase è necessaria per rendere gli elementi del corpus comparabili e può incrementare notevolmente le *performance* dei modelli successivi (Schmauder *et al.*, 2000). In particolare, dal corpus sono stati rimossi i segni di punteggiatura, i simboli, i numeri e i separatori mentre sono state preservate le parole composte tramite il trattino (-); tutti i termini sono stati convertiti in minuscolo. In seguito a questa pulizia è stato possibile costruire una *document-term matrix* (o *document-feature matrix*): in questo tipo di matrice sparsa, ogni riga corrisponde ad un documento del corpus ed ogni colonna a un termine distinto (chiamato *type*) presente nell'insieme dei 600 testi complessivi. I valori contenuti nelle celle della matrice possono essere le frequenze assolute dei termini, le frequenze normalizzate rispetto alle lunghezze dei singoli documenti, o pesi particolari come il *tf-idf*. Nell'ambito dell'*information retrieval* questo termine è l'abbreviazione di *term frequency - inverse document frequency*

ed è una misura molto diffusa (Salton *et al.*, 1988) che quantifica l'importanza di una parola in un documento, tenendo conto anche della sua frequenza nel corpus. Tale funzione aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento, ma cresce in maniera inversamente proporzionale con la frequenza del termine nel corpus. Il suo scopo è cogliere il linguaggio rilevante, cioè i termini che compaiono più volte in pochi documenti, per cui diventano caratteristici di quei testi.

2.2.3 Descrizione preliminare del corpus

Analizzando i valori della *document-feature matrix* (DFM) è possibile ottenere alcune statistiche di base sulla collezione di testi a disposizione. Il corpus contiene 250 738 parole (token) e ha un vocabolario di 17 296 parole distinte (*type*). I testi variano tra una lunghezza di 395 e 444 parole, non rispettando esattamente la richiesta di un range tra 400 e 440 parole. Tuttavia, dal momento che i numeri dipendono anche dal *software* di conteggio e da scelte interne ad esso, è stato ritenuto un risultato valido per proseguire con le analisi. Il corpus ha un *type-token ratio* (TTR) del 6.898% (ovvero una frequenza media di parola per *type* pari a 14.497) e una percentuale di *hapax legomena* pari al 45.6% del vocabolario (*hapax/type*), dimostrando un livello abbastanza alto di ridondanza. Il valore del TTR deve essere inferiore al 20% per poter condurre analisi accurate, mentre il numero di *hapax/type* deve essere inferiore al 50%, altrimenti il vocabolario risulta essere troppo originale (Tuzzi, 2005). Il corpus rispetta dunque i requisiti richiesti.

Ulteriori statistiche descrittive e i loro valori per i due corpus presi singolarmente possono essere osservate nella Tabella 2.1. Nell'ultima colonna viene riportato un *p*-value relativo all'ipotesi nulla che le distribuzioni dei valori nei testi riguardanti i due autori siano equivalenti, contro una generica ipotesi alternativa bilaterale. Nei casi in cui possono esserci differenze, è stato utilizzato il test non parametrico di Mann-Whitney visto che non ci si aspetta né normalità delle distribuzioni né omoschedasticità.

Analizzando i dati presenti nella Tabella 2.1 è possibile fare alcune considerazioni preliminari sulle differenze tra i due corpus. Coerentemente con le attese, il numero di token per testo, la loro deviazione standard, il loro massimo, il loro minimo e il numero di token totali non variano significativamente tra le due produzioni linguistiche. Questo per via di come è stato costruito il corpus: in modo tale che i risultati fossero standardizzati e comparabili per i due autori. Diversamente, osservando il numero di *type* si può affermare che il vocabolario dei testi umani sia più vasto, dal momento che contiene un numero maggiore di parole con frequenza unitaria. Il TTR è conseguentemente più alto per il corpus "Human" dato che il denominatore del rapporto è nei due casi molto simile.

Tabella 2.1: Statistiche descrittive di base sui corpus a disposizione.

Statistiche descrittive	**AI**	**Human**	**Totale**	***p*-value**
N testi	300	300	600	/
N token	125 291	125 447	250 738	/
SD degli N token	12.669	13.543	13.105	/
Max token	444	441	444	/
Min token	395	395	395	/
Token medi per testo	417.637	418.157	417.897	/
N *type*	9639	13 504	17 296	<2.2e-16
TTR	7.693	10.765	6.898	<2.2e-16
Hapax/type	0.423	0.512	0.456	9.033e-15
Frasi medie per testo	18.780	13.716	16.248	<2.2e-16
Token medi per frase	22.238	30.485	25.719	<2.2e-16

Entrambi i valori si collocano abbondantemente sotto la soglia prestabilita per il valore dei TTR. Inoltre, il rapporto tra *hapax* e *type* è maggiore nel caso dei testi prodotti da umani, anche se esso supera di poco il limite del 50% per l'originalità del vocabolario. Siccome il valore di questa soglia è ricavato empiricamente e può variare a seconda del tipo di testo e del contesto, si è deciso di proseguire ugualmente con le analisi. Infine, si può notare come i testi prodotti da ChatGPT contengano mediamente più frasi e, come conseguenza diretta, che esse abbiano lunghezza minore.

Le prime conclusioni che si possono trarre riguardano le differenze tra i due stili di scrittura. I testi scritti da esseri umani tendono a utilizzare una varietà più ampia di parole, riflettendo probabilmente una maggiore creatività e diversità nell'uso della lingua. Anche se i testi generati dall'intelligenza artificiale contengono molte parole, queste tendono a ripetersi più frequentemente. Verosimilmente, il risultato rispecchia la natura probabilistica di LLM come GPT, cioè di determinare la parola più probabile dato un prompt e/o una porzione di testo, come descritto nel Capitolo 1. Un TTR più alto nei testi umani indica una maggiore densità lessicale, ovvero una maggiore proporzione di parole uniche rispetto al totale delle parole. Questo è un segno di ricchezza linguistica e complessità che si riduce nei testi prodotti da ChatGPT. Il rapporto *hapax/type* suggerisce che gli esseri umani tendono a creare frasi uniche e ad utilizzare parole rare con maggiore frequenza piuttosto che riutilizzare termini noti e sicuri come nel caso del LLM. La lunghezza delle frasi potrebbe indicare una maggiore complessità nella sintassi e una più sicura capacità di articolare idee complesse in una singola frase. Al contrario, la strategia di ChatGPT nell'utilizzare frasi brevi potrebbe essere dovuta a un obiettivo di

maggiore leggibilità e di riduzione degli errori sintattici. L'analisi evidenzia le differenze intrinseche e significative tra i due stili di scrittura che, nelle analisi successive, possono essere sfruttate in ottica di classificazione e di previsione dell'autore.

2.2.4 Analisi descrittive

Dopo aver descritto la costruzione e le proprietà di base del corpus, è opportuno approfondire ulteriormente le sue caratteristiche attraverso analisi descrittive più dettagliate. Queste analisi permettono di comprendere meglio la sua composizione, rilevando le principali tendenze e peculiarità presenti nei testi, basandosi su vari aspetti linguistici e stilistici.

Legge di Zipf

I corpus, divisi per autore, possono essere confrontati tramite la legge di Zipf (1949) secondo cui è nota la distribuzione delle parole in un vocabolario: la frequenza f ed il rango r sono inversamente proporzionali, per cui: $f * r = c$. In altre parole, la legge di Zipf suggerisce che ci sono poche parole che compaiono molto frequentemente, mentre la maggior parte delle parole appare molto raramente. È possibile confrontare quanto bene le distribuzioni empiriche della legge nei due corpus si adattano al modello teorico.

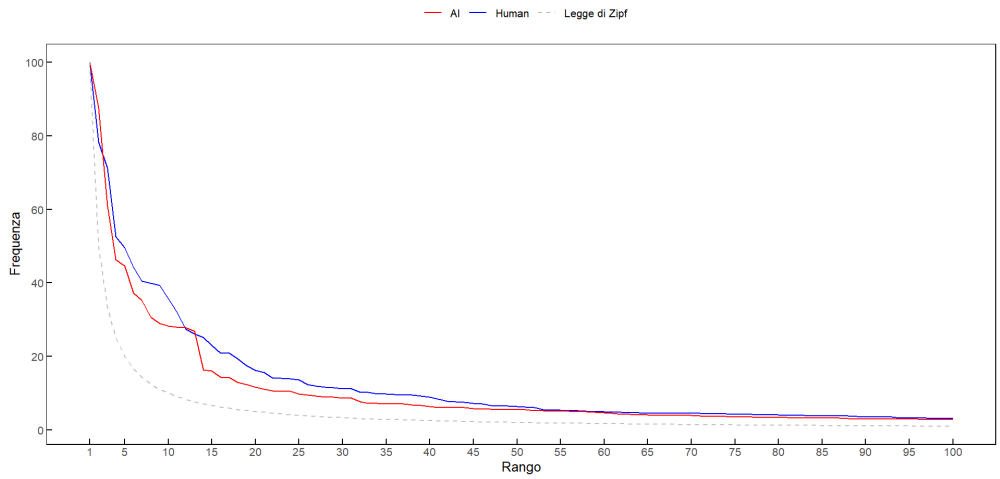


Figura 2.1: Adattamento teorico ed empirico della legge di Zipf nei due corpus. I valori delle frequenze sono stati normalizzati dividendoli per il valore massimo osservato per ciascuno dei due corpus e poi moltiplicati per cento. La linea tratteggiata mostra l'andamento teorico che segue la legge.

Il metodo di normalizzazione adottato in Figura 2.1 rende evidente quanto spesso una parola appaia rispetto alla parola più frequente, facilitando il confronto tra parole e corpus diversi. Per entrambi i corpus si denota un buon adattamento ai dati, in particolare nella parte iniziale e finale della curva. Non emergono particolari differenze nell'applicazione della legge per le due produzioni di testi.

Come diretta conseguenza, si osservano le frequenze assolute delle dieci parole più frequenti nei due corpus. Prima di eseguire questa analisi è necessario applicare una modifica alla codifica delle parole nei testi. Si decide di sostituire tutti gli apostrofi (caratteri ` ' ') con degli spazi, in modo che il conteggio delle parole possa avvenire correttamente, a prescindere dalla presenza di un'elisione davanti ad esse. Alcuni casi particolari dove l'apostrofo va mantenuto sono stati trattati singolarmente.

Tabella 2.2: Confronto delle frequenze assolute dei dieci token più presenti nei due corpus.

	AI		**Human**	
Rango	**Token**	**Freq. Assoluta**	**Token**	**Freq. Assoluta**
1	e	5695	di	4796
2	di	4987	e	3755
3	la	3472	che	3416
4	un	2639	la	2522
5	che	2537	in	2379
6	il	2123	un	2117
7	in	2009	per	1944
8	le	1749	a	1914
9	l	1648	il	1886
10	è	1607	è	1709

Dalla Tabella 2.2 si può subito notare come le parole più frequenti siano molto simili: nello specifico, in 8 casi su 10 sono presenti in entrambi gli ordinamenti. Come ci si poteva aspettare, i token che compaiono più spesso rappresentano preposizioni semplici, congiunzioni o articoli, che sono tra le parti del discorso più diffuse nei testi. Il fatto che le *stopwords* siano così frequenti e simili tra i due corpus può indicare la necessità di eseguire una pre-elaborazione dei dati per rimuovere queste parole, prima di utilizzare i testi per la classificazione o altre analisi. Questa operazione può aiutare a ridurre il rumore nei dati e a concentrarsi sulle parole più informative, soprattutto se si utilizza un approccio *bag of words*. Infatti, le *stopwords*, come articoli, preposizioni e congiunzioni, spesso vengono rimosse nei processi di *text mining* e di elaborazione del linguaggio naturale poiché tendono a non aggiungere significato semantico utile per molte applicazioni. In

realtà, diversi studi (Riloff, 1995, Monsteller e Wallace, 1963) dimostrano l'importanza delle *stopwords* in ambito di attribuzione d'autore: possono riflettere lo stile di scrittura o il tono di un testo, possono contribuire a formare sequenze di parole informative e possono essere adoperate come indicatori di struttura del discorso. Nei capitoli successivi si analizzerà se la loro presenza sarà utile o meno a discriminare i due corpus.

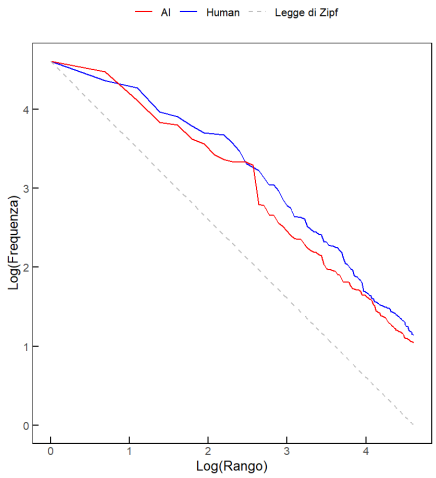


Figura 2.2: Adattamento teorico ed empirico della legge di Zipf nei due corpus, utilizzando la trasformazione logaritmo.

Una formulazione più recente della legge prevede: $\log(f) = c + a * \log(r)$. Il passaggio alla trasformazione logaritmica permette di utilizzare strumenti di regressione lineare per analizzare i dati. Un buon adattamento si verifica quando il coefficiente a assume valori vicini a -1. Adattando due modelli di regressione semplice alle frequenze normalizzate si ottengono i seguenti risultati.

Tabella 2.3: Coefficienti ed R^2 relativi al modello basato sulla legge di Zipf adattato ai dati.

	AI			**Human**		
	Coef	**St. error**	***p*-value**	**Coef**	**St. error**	***p*-value**
c	5.18	0.0398	<2e-16	5.41	0.0551	<2e-16
$\log(r)$	-0.901	0.0106	<2e-16	-0.910	0.0147	<2e-16
R^2		0.986			0.975	

L'attenzione è posta sui coefficienti relativi alla pendenza a che presentano valori molto vicini a -1, anche se questo valore non è compreso nell'intervallo di confidenza

dei parametri. I dati seguono abbastanza bene la relazione prevista dalla legge di Zipf come si osserva nella Figura 2.2; il lieve scostamento per il parametro a dal valore teorico pari a -1 può essere accettabile in molti contesti, specialmente in questo ambito empirico dove è comune osservare lievi variazioni. I valori molto elevati di R^2 mostrano un buon adattamento dei modelli ai dati, rafforzando questa interpretazione e indicando l'elevata adeguatezza della distribuzione prevista.

Perplexity

La *perplexity* è una funzione di probabilità considerata tra le più importanti metriche nell'ambito dell'NLP (Meister e Cotterel, 2021). Essa costituisce uno dei principali campanelli d'allarme per discriminare i testi. La perplessità di un modello linguistico riguardo ad un insieme di parole è data dall'inverso della probabilità di questo insieme, normalizzato per il numero di parole.
Per un insieme di parole $W = w_1, w_2, w_3, \ldots, w_N$

$$perplexity\,(W) = P\,(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P\,(w_1 w_2 \ldots w_N)}}$$

Oppure, utilizzando la regola della catena per espandere la probabilità di W:

$$perplexity\,(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Più alta è la probabilità del *set* di parole, più bassa sarà la perplessità. Un valore più basso di perplessità indica una maggiore fiducia del modello nelle sue previsioni. Un testo originale redatto da un essere umano tende a essere meno strutturato e più imprevedibile, rispetto a un testo generato da un'AI che presenta un punteggio di perplessità più basso. È possibile verificare ciò, addestrando un modello linguistico su una parte del corpus di testi prodotti dagli studenti, e poi calcolando la perplessità sulla porzione di testi rimanenti, utilizzando il metodo della convalida incrociata. In seguito, si procede in modo analogo per i testi prodotti da ChatGPT.

Servendosi di un modello linguistico basato sugli n-grammi e sul metodo di lisciamento *Kneser-Ney*, si può notare che i testi prodotti dagli studenti abbiano mediamente valori di *perplexity* più elevati rispetto ai testi prodotti dal modello linguistico, come evidenziato dalla Figura 2.3. Questo risultato può essere attribuito a diversi fattori e ci sono varie motivazioni che meritano attenzione.

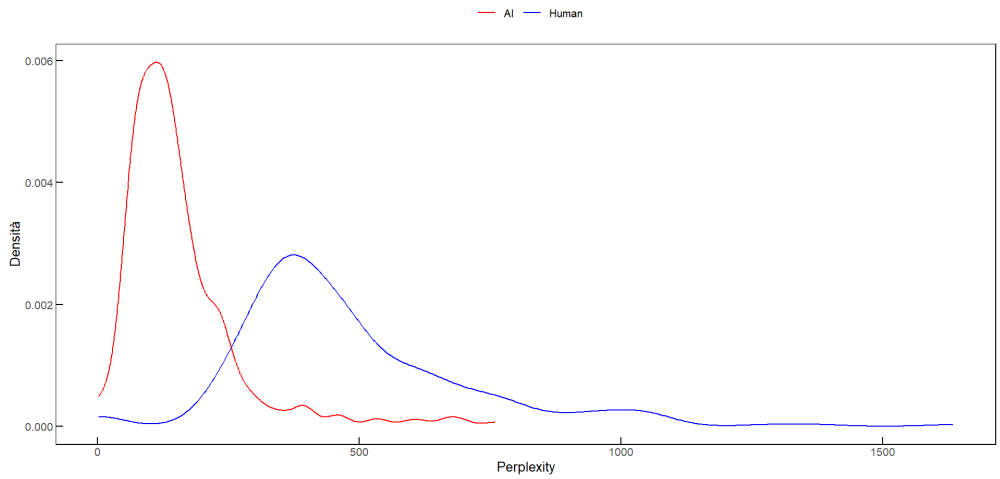


Figura 2.3: Densità dei valori di *perplexity* associati ai testi nei due corpus.

In primo luogo, il linguaggio umano tende ad essere più complesso e vario rispetto al linguaggio generato da un LLM. I modelli di linguaggio, soprattutto quelli basati sugli n-grammi, sono addestrati su dati e cercano di catturare le regolarità nelle sequenze di parole. I testi umani possono contenere espressioni idiomatiche, gerghi, variazioni stilistiche e sintattiche che sono meno prevedibili per il modello, risultando in una maggiore *perplexity*. Gli algoritmi di generazione del linguaggio tendono a produrre frasi che sono grammaticalmente corrette ma che possono essere meno ricche in termini di diversità lessicale e complessità sintattica. Questa semplicità può ridurre la *perplexity*.

Inoltre, se il modello di linguaggio è stato addestrato su testi simili a quelli generati dall'AI, potrebbe essere meglio adattato a prevedere questi testi. I testi generati artificialmente potrebbero essere più simili ai dati di addestramento, usando strutture e parole che il modello già conosce bene. Anche se il lisciamento *Kneser-Ney* è uno dei migliori metodi di lisciamento per i modelli basati su n-grammi (Kneser e Ney, 1995), può ancora avere difficoltà con le sequenze di parole rare o inaspettate che sono più comuni nei testi umani. Questo perché il lisciamento tenta di ridistribuire la probabilità ai contesti non visti, ma potrebbe non essere sufficiente per catturare tutta la variabilità del linguaggio umano.

Di conseguenza, i testi umani risultano avere una *perplexity* più alta perché sono generalmente più complessi e variati rispetto ai testi generati dall'AI, che tendono ad essere più prevedibili e meno diversificati.

Le parti del discorso (POS)

Un'altra analisi tipica del confronto tra testi riguarda la frequenza con cui compaiono le varie parti del discorso (POS, *part of speech*). Raggruppare le parole presenti nel testo, secondo la categoria grammaticale a cui appartengono, può dare un'indicazione sulla struttura delle frasi e del modo in cui vengono costruite.

Servendosi di modelli pre-addestrati per la lingua italiana, è possibile ottenere il numero di nomi, verbi, aggettivi, avverbi e pronomi nei due corpus e analizzarne le differenze. Per effettuare questa classificazione sono state utilizzate delle funzioni apposite contenute nella libreria `udipipe` (Straka *et al.*, 2017) di `R` che si basa sui lavori di ricerca condotti dall'Istituto di Linguistica Formale e Applicata (UFAL), dell'università di Praga.

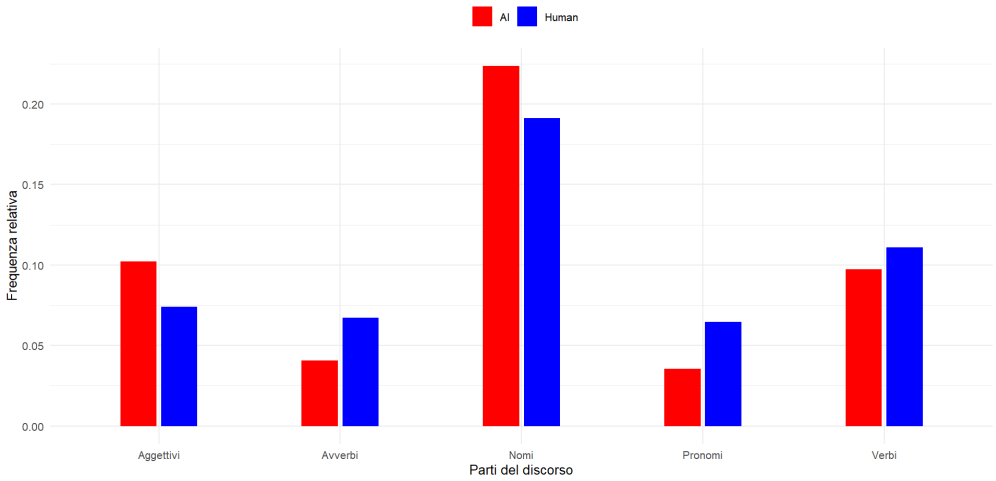


Figura 2.4: Frequenze relative delle principali parti del discorso all'interno dei corpus.

I valori nella Figura 2.4 mostrano le frequenze relative di alcune classi lessicali distinte per autore, all'interno dei due corpus. La scelta è ricaduta sulle classi più frequenti e più comuni in un testo, in modo da poter effettuare un confronto più robusto. Dai risultati emerge che i modelli linguistici adoperano nomi e aggettivi in maggiore quantità, mentre gli studenti usano più spesso verbi, avverbi e pronomi.

Una prima spiegazione a questo risultato può essere che gli algoritmi di generazione del linguaggio, specialmente quelli addestrati su grandi corpus di testo, spesso producono testi che sono più descrittivi e informativi, utilizzando molti nomi e aggettivi per costruire frasi dettagliate e precise. Questo può dipendere dal fatto che tali parole sono

frequentemente utilizzate nei dati di addestramento, che possono includere articoli, saggi e altre forme di scrittura formale.

I modelli linguistici tendono a focalizzarsi sulla creazione di descrizioni specifiche e dettagliate, utilizzando nomi e aggettivi per fornire informazioni chiare e comprensibili. Questo approccio aiuta a garantire che il contenuto generato sia facile da comprendere e rilevante per il contesto. Le frasi generate sono spesso strutturalmente semplici e dirette. In questo contesto, i nomi e gli aggettivi sono fondamentali per costruirle. Al contrario, l'uso di avverbi, pronomi e verbi può essere più complesso e variegato, richiedendo una comprensione più profonda delle sfumature del linguaggio naturale.

I testi umani tendono ad essere più ricchi e diversificati in termini di uso della lingua. Gli esseri umani utilizzano uno spettro più ampio di strutture sintattiche, includendo avverbi, pronomi e verbi per esprimere azioni, stati d'animo, relazioni temporali e altre sfumature. Questa varietà riflette la complessità del pensiero umano. Inoltre, nei testi umani, specialmente in contesti narrativi, l'uso di pronomi e verbi è essenziale per mantenere il flusso della comunicazione e per creare connessioni tra le idee. Gli avverbi sono spesso utilizzati per aggiungere sfumature e dettagli alle azioni e alle descrizioni, rendendo il testo più dinamico e coinvolgente.

Alla luce di queste considerazioni, le differenze nelle frequenze relative delle parti del discorso tra i due corpus possono essere spiegate dalle diverse priorità e strutture dei due tipi di testo. I testi prodotti da un modello linguistico tendono ad essere più descrittivi e specifici, con un uso maggiore di nomi e aggettivi, mentre i testi umani sono più variati e dinamici, con un uso maggiore di avverbi, pronomi e verbi, riflettendo la complessità e la ricchezza del linguaggio naturale umano.

Tabella 2.4: Frequenza assoluta e relativa per diverse categorie grammaticali nei testi AI e Human, con relativo p-value.

	Frequenza assoluta		**Frequenza relativa**		
	AI	**Human**	**AI**	**Human**	***p*-value**
Nomi	33 351	28 484	0.224	0.191	<2.2e-16
Verbi	14 493	16 517	0.0971	0.111	<2.2e-16
Aggettivi	15 260	10 036	0.102	0.0740	<2.2e-16
Avverbi	6046	9995	0.0405	0.0670	<2.2e-16
Pronomi	5313	9628	0.0356	0.0646	<2.2e-16

Occorre però verificare se le differenze tra le frequenze relative nei due casi sono significative. Per fare ciò è possibile confrontare i valori delle frequenze per i vari POS nei due corpus e valutare la significatività delle differenze tramite un test per le proporzioni.

Il test si basa sulla statistica *Chi-quadrato* e valuta l'ipotesi nulla che i valori di due proporzioni non siano significativamente differenti, contro una generica ipotesi alternativa bilaterale, come riportato nella Tabella 2.4.

Dal momento che tutti i p-value sono minori del valore soglia $\alpha = 0.05$ si può concludere che le differenze per le frequenze relative di ogni tipo di POS nei due corpus sono significative. Inoltre, applicando un *Test di Indipendenza Chi-quadro* sulle frequenze assolute prese congiuntamente nei due casi, si può stabilire che c'è una differenza significativa nelle distribuzioni dei POS tra i due corpus (p-value inferiore a 0.05).

Sentiment ed emozioni

Infine, la polarità di un testo può essere un ulteriore indicatore dell'identità del suo autore. Attraverso dei modelli pre-allenati, e utilizzando appositi dizionari ontologici, è possibile assegnare un valore alla polarità di ciascuna frase in ogni testo. Le parole in ogni frase vengono cercate e confrontate con un dizionario di parole polarizzate, al fine di contrassegnarle con un valore numerico. Parole positive vengono associate a valori maggiori di zero e parole negative a valori minori di zero. Attraverso la somma dei valori associati ad ogni parola, si può calcolare un contributo alla polarità negativa e positiva per ogni frase, da cui è possibile ottenere un valore approssimativo del *sentiment* di un testo.

Dal momento che i corpus sono stati creati rispondendo alle stesse domande, relative alle stesse tematiche, può essere opportuno valutare se ci siano differenze significative nella polarità delle due collezioni di testi. In questo caso è stata utilizzata la libreria `syuzhet` di `R` che si basa sul *software* CoreNLP, sviluppato dall'NLP Group dell'Università di Stanford (Manning *et al.*, 2014). Nello specifico, è stata utilizzata la versione del dizionario *NRC Word-Emotion Associated Lexicon*[2] specifica per la lingua italiana (Socher *et al.*, 2013).

Per effettuare le analisi al meglio è necessario adottare alcune accortezze sul peso delle diverse parole all'interno del corpus visto che, in ambito di *sentiment analysis* riguardo ai testi, è nota anche la presenza del *Pollyanna Effect*. Conosciuto anche come *Pollyanna Principle*, è un fenomeno psicologico che si osserva quando le persone tendono a utilizzare un linguaggio più positivo rispetto a uno negativo (Matlin *et al.*, 1978). In altre parole, gli esseri umani hanno una predisposizione a ricordare, interpretare e comunicare informazioni in modo positivo. Nell'analisi delle polarità, questo effetto si riferisce alla tendenza di un corpus di testo a contenere più parole positive rispetto a

[2]Consultabile al sito: http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm.

quelle negative. Non tenerne conto può portare a una stima errata del *sentiment*, visto che un documento potrebbe sembrare più positivo di quanto non sia realmente a causa della prevalenza di parole di questo tipo.

Per questo motivo si è deciso di classificare il *sentiment* di un testo come positivo nei casi in cui la somma delle polarità positive al suo interno superasse il 60% del valore complessivo. In modo equivalente, i contributi delle frasi alla polarità negativa sono stati cambiati di segno e moltiplicati per 1.5. Così facendo, un testo viene classificato come positivo solo se il suo valore complessivo risulta essere maggiore di zero, e si riesce a dare un peso maggiore alle parole o frasi negative.

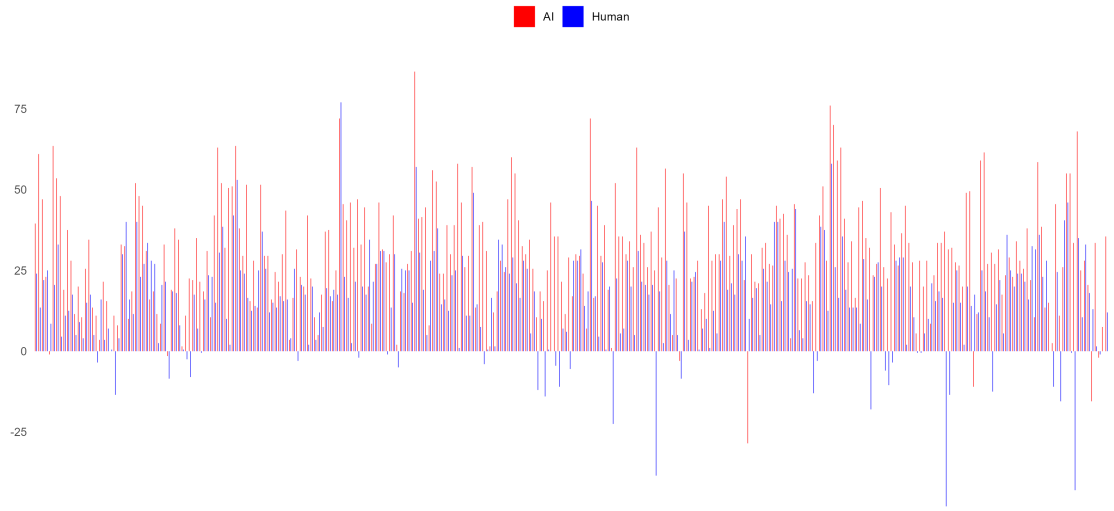


Figura 2.5: Polarità di ogni testo nel corpus complessivo.

Osservando la Figura 2.5 appare subito evidente come la quasi totalità dei testi abbia polarità positiva. Questo dipende principalmente dalle tematiche affrontate all'interno di essi, che prevedono situazioni in cui è difficile attribuire connotazioni negative ai discorsi e utilizzare termini con polarità minore di zero. D'altro canto, i pochi testi classificati con i valori di *sentiment* negativi appartengono quasi tutti al corpus prodotto dagli studenti. Questo risultato fornisce un'indicazione rilevante sulla scelta delle parole da utilizzare quando il modello linguistico crea del testo. Basandosi su questa classificazione è più probabile che parole con connotazione negativa vengano adoperate dagli studenti, piuttosto che dal LLM tramite il quale è stata condotta l'analisi. A sostengo di questa tesi, si può notare come i testi prodotti da ChatGPT siano associati, la maggior parte delle volte, a valori di polarità più alti rispetto ai corrispettivi testi scritti da umani.

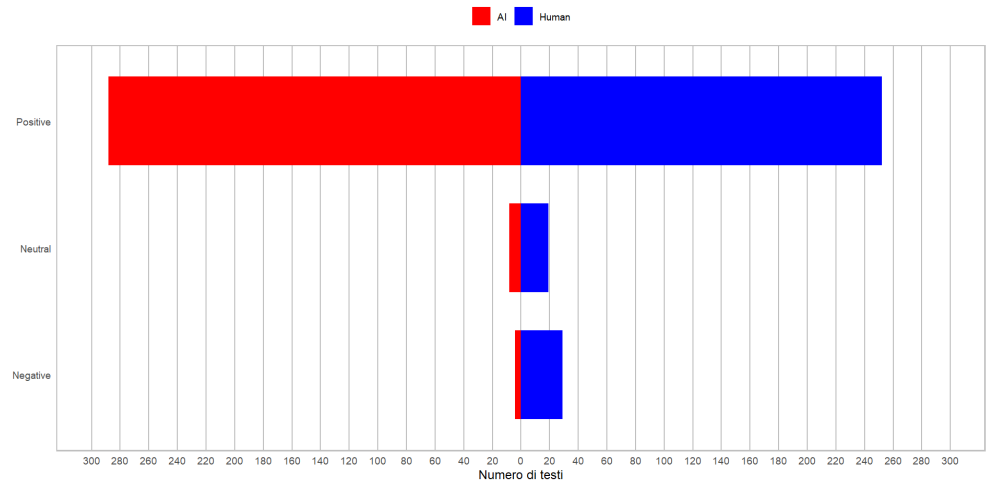


Figura 2.6: Classificazione dei testi in tre categorie relative al *sentiment*.

Applicando una soglia arbitraria (*treshold*), è possibile suddividere i testi in base alla loro polarità in tre distinte categorie: "*positive*" per i testi con polarità superiore a *treshold*, "*neutral*" per testi con valori di polarità compresi tra -*treshold* e *treshold* e "*negative*" per testi con polarità inferiore a -*treshold* e osservare la loro numerosità nei due corpus. Un approccio comune è quello di definire una soglia basata su una frazione (un decimo) dello scarto interquartile dei valori del *sentiment* nei due corpus. In questo modo si utilizza un valore diverso per le due tipologie di testi, che si adatta meglio alle loro caratteristiche. Sebbene la scelta della soglia condizioni i risultati, della Figura 2.6 appare evidente che il maggior numero di testi ricada nella categoria "*positive*" e che questa sia più numerosa per la produzione di testi appartenente a ChatGPT, come sottolineato in precedenza. Le categorie "*neutral*" e "*negative*" sono più esigue e contengono al loro interno, per la maggior parte, testi prodotti dagli studenti.

Analizzando nella Tabella 2.5 i testi a due a due, si possono notare le differenze nella classificazione dei testi con lo stesso tema ma appartenenti ad autori differenti. Delle 300 coppie di testi, 251 sono stati classificati con la stessa polarità. È raro che un testo scritto da AI sia classificato come negativo e che lo stesso tema abbia un *sentiment* positivo se scritto da un umano, mentre è più comune che si verifichi il contrario.

Queste analisi mettono in luce ulteriori aspetti che consentono di effettuare alcune considerazioni sulle differenze stilistiche dei due autori. Gli algoritmi dei modelli linguistici possono avere difficoltà a generare testi con una polarità fortemente negativa. La presenza di testi con polarità negativa, quasi tutti scritti dagli studenti, potrebbe indi-

Tabella 2.5: Tabella a doppia entrata dei testi in base alla loro polarità nei due corpus. Nei totali di riga e di colonna sono contenuti i valori marginali delle tre categorie per i due corpus di testi.

Human \ AI	**Positivo**	**Neutro**	**Negativo**	**Totale**
Positivo	247	2	3	252
Neutro	16	3	0	19
Negativo	25	3	1	29
Totale	288	8	4	300

care che ChatGPT tende a produrre testi più neutrali o positivi, forse a causa di una mancanza di comprensione delle sfumature emotive o per evitare di generare contenuti inappropriati. Molti modelli di linguaggio sono progettati per essere neutri o positivi per eludere problemi come la generazione di contenuti offensivi o negativi. Gli sviluppatori spesso utilizzano filtri e strategie per limitare l'uso di parole con connotazione negativa, il che può portare a testi con polarità tendenzialmente positiva. I testi generati dall'intelligenza artificiale tendono ad avere un tono e uno stile più neutri o ottimistici per garantire la loro accettabilità e utilità in vari contesti.

Gli esseri umani, d'altra parte, possono modulare il loro tono e stile in base alla situazione, risultando in una varietà più ampia di accezioni. Bisogna tenere in considerazione anche che molti modelli di analisi del *sentiment* hanno difficoltà a rilevare ironia e sarcasmo, spesso presenti nei testi umani. Ciò può portare a una stima distorta della polarità nei testi umani rispetto ai testi generati da AI, che tendono ad essere più diretti. Il *Pollyanna Effect* può influenzare sia un corpus di testi scritto da umani sia uno scritto da un modello linguistico, che riflette la natura dei suoi dati di addestramento. L'entità e la tipologia dell'effetto possono variare tra i due tipi di testi e, in questo caso, si può tenerne conto a livello di interpretazione dei risultati, come una delle motivazioni che portano alla grande disparità di numerosità tra i diversi *sentiment* nei corpus.

Dallo stesso dizionario è possibile ricavare le emozioni presenti nei testi. Per come è strutturato il *Lexicon* (Mohammad e Turney, 2013), ad ogni parola possono essere associate una o più emozioni tra *anger*, *fear*, *anticipation*, *trust*, *surprise*, *sadness*, *joy*, e *disgust*. Questa suddivisione denominata *Plutchik's wheel of emotions* considera otto emozioni primarie in cui, per ognuna di esse, è presente il suo opposto (Plutchik, 1980). In modo analogo a quanto fatto per le polarità, è possibile ottenere i valori associati ad ogni emozione per ogni frase e assegnare al testo l'emozione più presente. Tuttavia, dal momento che nel *Lexicon* le liste di parole associate ad ogni emozione hanno lunghezze

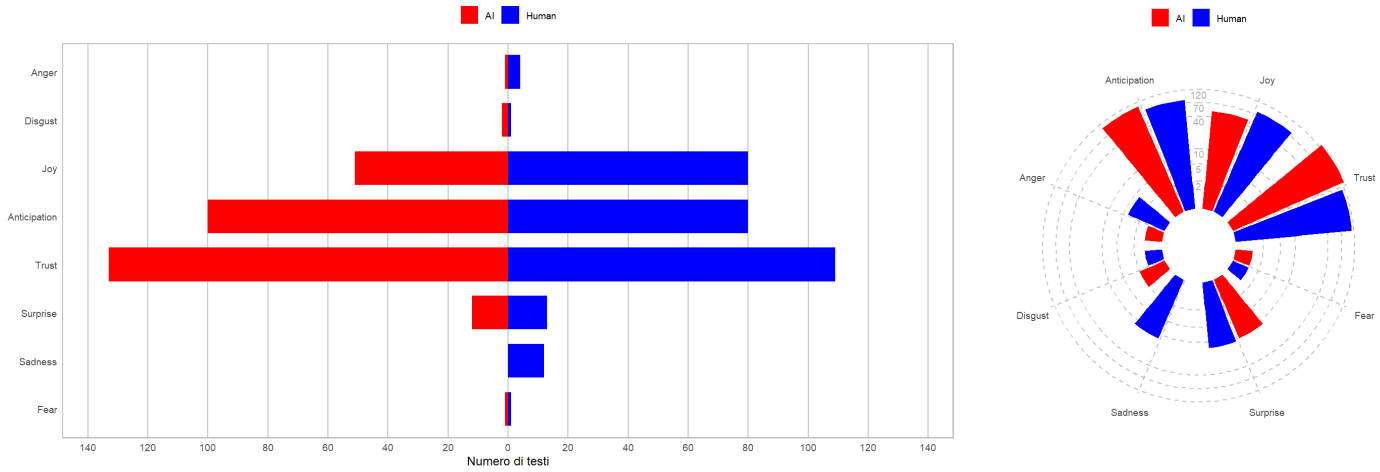


Figura 2.7: Classificazione dei testi in base all'emozione più presente (a sinistra) e *Plutchik's wheel of emotions* (a destra). La lunghezza delle barre nella ruota riflette i valori in scala logaritmica mentre le etichette riportano i valori su scala originale; le emozioni sono posizionate in modo da evidenziare le loro relazioni opposte.

differenti (Mohammad e Turney, 2010), si è ritenuto più corretto utilizzare le frequenze relative delle emozioni, standardizzate per le lunghezze delle relative liste.

Dalla Figura 2.7 si può notare che i testi nel corpus sono associati principalmente ad emozioni positive come fiducia, gioia e aspettativa (viene inserita come l'opposto di sorpresa), mentre rabbia, disgusto, tristezza e paura sono poco presenti. Non emerge una particolare differenza tra le suddivisioni nei due corpus ma è comunque possibile trarre qualche considerazione. Non sono presenti testi associati a tristezza per il corpus AI, mentre ciò accade più spesso per i testi prodotti da studenti. Anche i testi associati alla rabbia sono più presenti tra gli studenti, contrariamente a quanto avviene per disgusto e paura dove si distribuiscono in maniera abbastanza equivalente. La fiducia è l'emozione più presente in entrambi i corpus e questo dipende sia dalla natura dei temi su cui sono stati scritti i testi, sia dalla numerosità particolarmente alta di parole presenti nella relativa lista, nonostante la correzione effettuata per bilanciare questo aspetto.

Questi risultati sono coerenti con quelli relativi al *sentiment* dei testi che vede la polarità positiva come la categoria più presente. I testi prodotti dagli studenti sono più spesso associati a *sentiment* negativo e ciò rispecchia una maggiore frequenza di emozioni come rabbia e tristezza.

Dalle analisi emerge che la maggiore polarità positiva dei testi generati artificialmente e la presenza di testi con *sentiment* negativo quasi esclusivamente tra quelli prodotti dagli

studenti, riflettono differenze significative tra i due tipi di autori. Queste diversità sono confermate dallo studio sulle emozioni e possono avere implicazioni importanti per la comprensione, la classificazione e l'uso dei testi nei vari contesti applicativi. È importante notare che i risultati sono strettamente legati ai dati e al dizionario utilizzato che, trattandosi di una traduzione dalla lingua inglese, presenta per sua natura delle imprecisioni. Pertanto, è difficile estendere queste considerazioni ad un contesto più ampio.

2.3 Metodi di vettorizzazione

In seguito, vengono introdotte tre differenti metodologie per rappresentare i testi appartenenti al corpus. La prima, denominata *feature extraction*, prevede la conversione di dati di testo grezzi in un formato che può essere facilmente elaborato da algoritmi di apprendimento automatico. Questo processo può essere eseguito in diversi modi, ciascuno adattabile al contesto specifico in cui viene applicato. La seconda metodologia si basa su vettori costruiti da un modello linguistico pre-allenato, in grado di generare rappresentazioni contestualizzate dei testi chiamate *embedding*. Infine, è possibile utilizzare l'Analisi delle Corrispondenze per proiettare i testi sugli assi principali, in modo da poter ridurre la loro dimensione e valutare meglio le loro relazioni nel corpus.

Le rappresentazioni derivanti da queste tecniche verranno utilizzate nel capitolo successivo, in ambito di classificazione, per confrontare le loro capacità.

2.3.1 Feature Extraction

La *feature extraction*, o estrazione delle caratteristiche, è una tecnica fondamentale per ricavare dai testi informazioni che verranno poi usate nell'addestramento dei modelli. La scelta di queste *feature* ha un impatto significativo nelle prestazioni successive siccome alcune caratteristiche possono essere più rilevanti e informative di altre, in determinate circostanze. Negli anni, in letteratura sono state proposte una varietà di caratteristiche linguistiche che, se combinate, sono in grado di creare un profilo capace di identificare in modo univoco l'identità dell'autore. Lo stile di scrittura è un fenomeno complesso e sfaccettato, per questo motivo le strategie più efficaci tendono a combinare caratteristiche che rappresentano aspetti diversi e complementari della struttura linguistica (Mikros *et al.*, 2023).

Un particolare gruppo di *feature*, che si è dimostrato essere particolarmente rilevante nella classificazione dei testi, prende il nome di *Author's Multilevel Ngram Profiles* (AMNP). L'AMNP è stato proposto per la prima volta da Mikros e Perifanos (2013) ed è

un metodo di rappresentare un documento utilizzando n-grammi di grandezza crescente sia per le parole e sia per i caratteri. Le parole più frequenti tendono ad individuare lo stile individuale di un autore e sono facilmente interpretabili. L'analisi dei bi-grammi e tri-grammi può rivelare *pattern* sintattici e frasi ricorrenti che non sono evidenti quando si considerano solo le singole parole. Inoltre, catturano sequenze di termini, fornendo informazioni sul contesto e sulla coerenza linguistica. Infine, le sequenze di caratteri (i char-grammi) catturano informazioni dettagliate su ortografia, punteggiatura, prefissi e suffissi, che possono essere distintive dello stile di un autore e forniscono un livello di dettaglio più fine rispetto alle altre *feature*. L'analisi combinata di parole e n-grammi frequenti permette di ottenere una visione a più livelli del testo, cogliendo sia aspetti generali che dettagli specifici (Selj *et al.*, 2003).

In aggiunta, possono essere usate *feature* stilometriche di varia natura (tipologia e lunghezza delle parole, indici di diversità e complessità lessicale, ...) che, se affiancate a tecniche più robuste come l'AMNP, sono in grado di incrementare l'accuratezza dei risultati di un modello di classificazione.

Tabella 2.6: Elenco delle *feature* estratte dai testi, in base a tipologia e numerosità.

Tipologia	**N. *feature***	**Descrizione (numero di *feature*)**
Feature lessicali – basate sui caratteri	4	Numero di caratteri (1), numero medio di caratteri per frase (1), numero medio di caratteri per token (1), SD del numero di caratteri per token (1)
Feature lessicali – basate sulle parole	7	Numero medio di parole per frase (1), SD del numero medio di parole per frase (1), lunghezza massima frase (1), TTR (1), numero di *stopwords* (1), numero di parole con più di 7 caratteri (1), numero di parole con più di 10 caratteri (1)
Feature lessicali – basate sulla grammatica	9	Numero di pronomi personali (1), numero di congiunzioni (1), numero di token con genere (2), frequenze relative POS (5)
Metriche di leggibilità	2	Numero medio di sillabe per parola (1), indice Gulpease (1)
Indici di complessità e diversità lessicale	3	*Perplexity* (1), indice di Yule (1), *burstiness* (1)

Un elenco delle *feature* che è possibile utilizzare nelle analisi successive è presente nella Tabella 2.6. Si è scelto di utilizzare solo *feature* che in qualche modo aiutino a discriminare i due autori, evitando di inserire caratteristiche come il numero di token per

testo, che non sono significativamente diverse nelle due tipologie di testi.

Come descritto all'interno della tabella, oltre agli indicatori discussi in precedenza, si è deciso di tenere conto per ogni testo delle seguenti quantità: numero di caratteri, numero medio di caratteri per frase, numero medio di caratteri per token, SD del numero di caratteri per token, lunghezza massima di una frase, numero di *stopwords*, numero di parole con più di sette caratteri (*longwords*), numero di parole con più di dieci caratteri (*long-longwords*) numero di pronomi personali, numero di congiunzioni, numero medio di sillabe per parola. In aggiunta, sono stati utilizzati l'indice Gulpease, l'indice di Yule e un indicatore per la *burstiness*.

Il conteggio dei caratteri viene applicato ai testi originali e tiene conto di lettere, numeri, punteggiatura, spazi bianchi, altri simboli e caratteri speciali. In altre parole, è la lunghezza del testo misurata in caratteri, come apparirebbe in un editor di testo. Le *stopwords* sono parole comuni di una produzione linguistica, che non hanno a che vedere con un particolare argomento specifico. Sono parole che è possibile trovare in qualsiasi documento indipendentemente dall'argomento trattato. Esempi tipici sono articoli, preposizioni e congiunzioni. Non esiste un elenco definitivo che comprende tutte le *stopwords*, anche per via della loro variazione a seconda della lingua. In questo caso si è deciso di fare affidamento alla lista presente nella funzione `stopwords` appartenente alla libreria `quanteda` di `R`, che si basa sull'elenco redatto nel progetto *Snowball stemmer* (Boulton *et al.*, 2005). Anche per la definizione di "*longword*" non esiste un criterio universalmente riconosciuto, ma per la lingua italiana si è deciso di porre una soglia minima pari a otto caratteri. Inoltre, per rendere il confronto ancora più flessibile è stato conteggiato anche il numero di parole con più di dieci caratteri, chiamate in questo caso "*long-longwords*".

Per il conteggio di pronomi personali e congiunzioni si è deciso di utilizzare la classificazione adottata dalla funzione `udpipe_annotate` appartenente alla libreria `udipipe` di `R`. La stessa funzione è stata adoperata per individuare i token a cui è possibile assegnare un genere, all'interno della loro categoria grammaticale. In particolare, si è tenuto conto di nomi, aggettivi, pronomi, articoli e verbi (participi passati) nei casi in cui sono stati coniugati al maschile o al femminile. Riguardo al numero di sillabe per parola si è stabilito, per semplicità, di adottare una *proxy* che prevede di assegnare un valore pari al numero medio di vocali per parola. Questo metodo non è esatto ma a livello di singolo testo può fornire una buona approssimazione del vero valore del numero di sillabe.

L'indice Gulpease è un indice di leggibilità di un testo tarato sulla lingua italiana (Lucisano e Piemontese, 1988). Rispetto ad altri (Solnyshkina *et al.*, 2017) ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico. L'indice considera due variabili linguistiche: la lunghezza della parola

e la lunghezza della frase rispetto al numero delle lettere. La formula per il suo calcolo è la seguente:

$$89 + \frac{300 * (\text{numero di frasi}) - 10 * (\text{numero di lettere})}{\text{numero di parole}}$$

Usando il numero di lettere al posto del numero di caratteri non si tratta di una semplice combinazione di altre quantità già inserite nell'analisi. I risultati sono compresi tra 0 e 100, dove il valore 100 indica la leggibilità più alta e 0 la leggibilità più bassa.

L'indice di Yule è un indice di diversità lessicale che si propone di misurare la ricchezza del vocabolario di un testo (Rider e Yule, 1944) ed è utilizzato spesso per problemi come l'identificazione d'autore (Tanaka-Ishii *et al.*, 2015). Dato N il numero di parole in un testo, $V(N)$ il numero di parole distinte, $V(m, N)$ il numero di parole che appaiono m volte nel testo e m_{max} la più grande frequenza di una parola, l'indice di Yule è definito attraverso il momento primo e secondo della distribuzione della popolazione del vocabolario:

$$\begin{aligned} \mathrm{K} &= C\frac{S_2 - S_1}{S_1^2} \\ &= C\Big[-\frac{1}{N} + \sum_{m=1}^{m_{max}} V(m, N)\left(\frac{m}{N}\right)^2\Big] \end{aligned}$$

dove $S_1 = N = \sum_m mV(m, N)$, $S_2 = \sum_m m^2 V(m, N)$ e C è una costante definita da Yule pari a $C = 10^4$. L'indice è progettato per misurare la ricchezza del vocabolario di un testo: più grande è K, meno ricco è il vocabolario.

La *burstiness* è una particolare metrica adottata nell'ambito dell'NLP che misura l'imprevedibilità statistica e linguistica di un contenuto (Spagnuolo, 2023). Il termine, traducibile in italiano come scoppiettio, si riferisce alla tendenza di alcuni elementi linguistici (come parole specifiche) ad apparire in "raffiche" o in gruppi concentrati all'interno di un testo. La distribuzione non uniforme e a grappoli di tali elementi, va in contrasto con una distribuzione più regolare che ci si aspetterebbe in un testo scritto da un modello linguistico.

In letteratura (Irvine e Callison-Burch, 2017) sono presenti varie metriche che permettono di determinare questa quantità. Alcuni approcci (e.g. Pierrehumbert, 2012) la definiscono come la misura del picco di utilizzo di una parola in un particolare corpus di testi. Questa definizione identifica come parole che contribuiscono alla *burstiness*, quei token specifici che tendono ad apparire frequentemente in un documento quando viene trattato un determinato argomento, ma che non si presentano con la stessa frequenza in

tutti i documenti della raccolta. Tuttavia, questo metodo non permette di calcolare una misura specifica per ogni testo.

In alternativa, è possibile definire la *burstiness* di un testo t attraverso il rapporto tra la deviazione standard delle frequenze dei token al suo interno e la loro frequenza media:

$$Burstiness_t = \frac{\sigma_w}{\mu_w}$$

dove μ_w è calcolato a partire dalla DFM di ogni testo e indica la media delle frequenze di ogni token w, e σ_w indica la loro deviazione standard. Questo rapporto fornisce un'indicazione di quanto la variabilità delle frequenze delle parole è alta rispetto alla frequenza media. In altre parole, misura la dispersione relativa delle frequenze dei token nel testo. Valori alti di *burstiness* quantificano la tendenza delle parole ad avere una distribuzione molto variabile rispetto alla media, indicando che alcune parole possono apparire in modo molto irregolare o esplosivo in certe parti del testo. Ci si aspetta quindi che siano associati a testi umani.

Nella tabella 2.7 sono indicati i valori medi per i due corpus di ciascuna delle quantità menzionate, calcolate singolarmente su ogni testo. I risultati, basati su diverse metriche linguistiche, evidenziano differenze significative tra i testi scritti dai due autori.

I testi AI risultano avere più caratteri totali, ma meno caratteri per frase come ci si poteva attendere dai risultati precedenti. Infatti, le frasi più lunghe nei documenti, contengono in media più token nei testi scritti dagli studenti. Inoltre, il numero medio di caratteri per frase varia molto di più in questi ultimi, rispetto ai testi prodotti da ChatGPT. Al contrario, il numero medio di caratteri per token è più instabile nei testi AI, sebbene le parole al loro interno siano mediamente più lunghe. Questo indica la preferenza del modello linguistico nell'alternare parole di diverse dimensioni, prediligendo l'uso di quelle più lunghe.

Proseguendo, i testi umani contengono un numero significativamente maggiore di *stopwords*, ma minore di *longwords* e di sillabe per parola, suggerendo un vocabolario potenzialmente più complesso o tecnico da parte dell'intelligenza artificiale. I testi prodotti dagli studenti contengono un numero significativamente maggiore di pronomi personali, il che può indicare un approccio più diretto e personale nella scrittura, mentre il numero di congiunzioni sembra essere meno discriminante. Le due tipologie di testi non si differenziano per il numero di token al maschile al loro interno, mentre i testi prodotti da ChatGPT contengono, in media, un numero significativamente maggiore di token al femminile.

Secondo l'indice Gulpease, i testi scritti artificialmente risultano avere una leggibilità

Tabella 2.7: Valori medi delle *feature* nei testi, distinguendo per i due corpus. In aggiunta, viene riportato un p-value relativo all'ipotesi nulla che le distribuzioni dei valori per i due autori siano equivalenti, contro una generica ipotesi alternativa bilaterale. Dopo aver verificato che nessun gruppo di osservazioni segue una distribuzione Normale, il p-value è stato calcolato tramite il test non parametrico di Mann-Whitney per campioni indipendenti.

Quantità	**AI**	**Human**	***p*-value**
Numero di caratteri	2877.297	2686.920	<2.2e-16
Numero di caratteri per frase	155.730	217.240	<2.2e-16
Numero di caratteri per token	5.609	5.179	<2.2e-16
SD del numero di caratteri per token	3.437	3.107	<2.2e-16
SD del numero di parole per frase	7.086	14.325	<2.2e-16
Lunghezza massima frase	247.063	381.863	<2.2e-16
Numero di *stopwords*	174.460	185.430	<2.2e-16
Numero di *longwords*	132.650	98.670	<2.2e-16
Numero di *long-longwords*	42.103	27.457	<2.2e-16
Numero di pronomi personali	7.610	14.760	<2.2e-16
Numero di congiunzioni	27.870	26.493	0.00474
Numero di token al maschile	114.403	113.147	0.482
Numero di token al femminile	115.013	89.783	<2.2e-16
Numero medio di sillabe per parola	2.615	2.412	<2.2e-16
Indice Gulpease	46.574	47.371	0.00196
Indice di Yule	96.708	76.741	<2.2e-16
Burstiness	10.267	11.192	<2.2e-16

leggermente inferiore. Analizzando singolarmente i valori che lo compongono, si nota che il maggior responsabile di questo risultato è il numero di lettere nei testi, che è più elevato nel caso di quelli creati da ChatGPT. Il peso di questa caratteristica nell'indice non viene contrastato da quello del numero di frasi, seppur presenti in maggiore quantità nei testi AI. Siccome il numero di parole nei due corpus è molto simile, l'indice calcolato sui testi prodotti dagli studenti risulta essere in media maggiore. Entrambi i corpus presentano un valore di leggibilità medio, che secondo gli autori rientra nella categoria "testi difficili da leggere per chi ha la licenza media".

Per come è stato definito l'indice di Yule, la diversità del vocabolario dei testi AI è inferiore a quella dei testi umani, suggerendo che il modello linguistico potrebbe ripetere più spesso le stesse parole. Questo risultato è confermato anche dal maggior numero di *type* nel corpus Human. Come da aspettative, i testi prodotti da ChatGPT presentano valori di *burstiness* meno elevati. Questo accade probabilmente perché le frasi prodotte

artificialmente sono rigide e semplici nella struttura, mancando di creatività ed originalità. Di conseguenza, valori più alti di *burstiness* saranno associati a testi generati dagli studenti. Infatti, gli esseri umani tendono ad avere uno stile di scrittura più variabile e questo può portare ad utilizzare parole specifiche in modo molto concentrato in certe parti del testo. Al contrario, anche i modelli linguistici più avanzati, tendono a produrre testi con una distribuzione delle parole più uniforme e meno variabile.

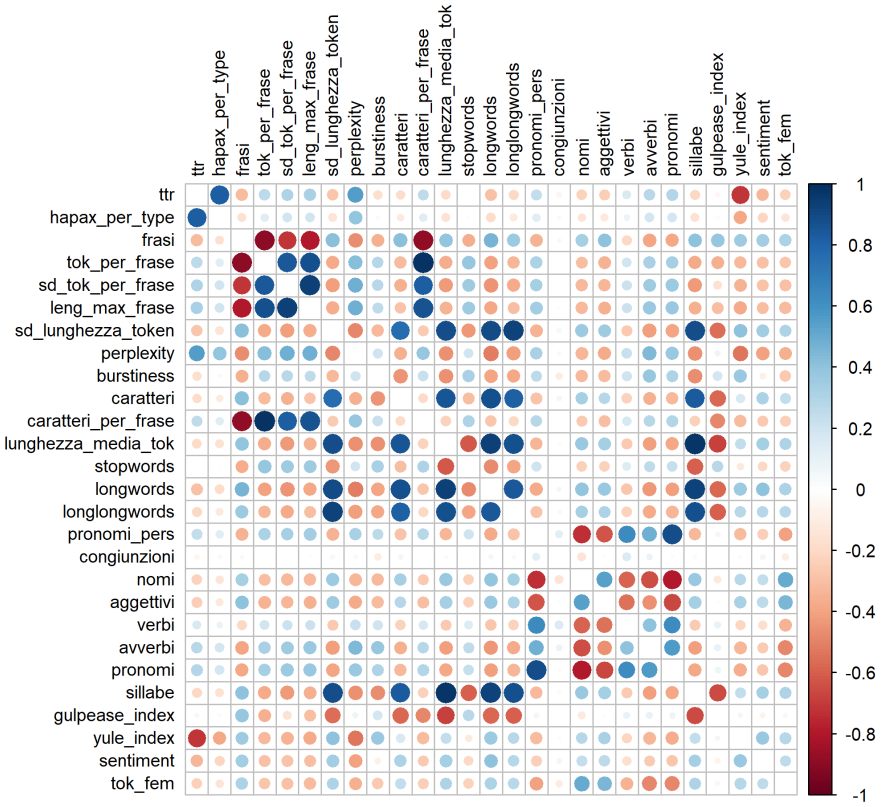


Figura 2.8: Matrice di correlazione per le Text Features.

La matrice di correlazione in Figura 2.8 mostra le relazioni tra le diverse variabili analizzate. In particolare, sono da sottolineare gli alti valori positivi tra `leng_max_frase` e `ttr`, e tra `longlongwords` e `sd_lunghezza_token`; come anche l'alta dipendenza lineare negativa tra `nomi` e `pronomi_pers`. In aggiunta, i tre principali indicatori di imprevedibilità del testo, *perplexity*, *burstiness* e `sd_tok_per_frase`, presentano una discreta correlazione positiva, anche se meno evidente di quanto ci si poteva attendere. Queste considerazioni evidenziano come le diverse caratteristiche linguistiche siano interconnesse, influenzandosi reciprocamente all'interno dei testi esaminati.

Inoltre, nonostante spesso si preferisca adoperare tecniche differenti, ci sono varie

ragioni per utilizzare un *set* di *feature* fatto a mano. Scegliendo accuratamente, infatti, esse risultano essere direttamente correlate al compito da svolgere e quindi maggiormente informative. Queste osservazioni possono avere implicazioni significative per il riconoscimento automatico, avendo dimostrato la significatività delle loro differenze nei due corpus. Il loro utilizzo, inoltre, permette grande flessibilità nel tipo di caratteristica da rilevare nel testo, al contrario di quanto è possibile ottenere basandosi su tecniche più standard come il *tf-idf*. Il risultato è un approccio più specifico che spesso porta a risultati migliori (Shijaku *et al.*, 2023). Un ulteriore aspetto da tenere in considerazione è l'interpretabilità dei risultati. Spesso tecniche più complesse e avanzate devono scontrarsi con il problema di comunicare i risultati delle analisi ad un pubblico non sempre esperto della materia in questione. Solitamente, la maggior parte delle *feature* risulta essere comprensibile e facilmente interpretabile.

2.3.2 Riduzione della dimensionalità

L'Analisi delle Corrispondenze (CA; Greenacre, 1984) è una tecnica di analisi statistica che identifica la struttura di dipendenza interna a una tabella di contingenza. Essa mira a determinare la struttura dell'eventuale dipendenza tra le modalità poste sulle righe e quelle poste sulle colonne di una matrice non necessariamente simmetrica. Lo scopo della CA è individuare un sistema di assi ortogonali che massimizzi la distanza *Chi-quadrato* tra le proiezioni dei profili (cioè le frequenze relative per riga e per colonna). Gli assi vengono ordinati in base alla loro importanza, determinata dai valori singolari (e quindi dalla varianza spiegata) ricavati dalla decomposizione SVD della tabella di contingenza. Utilizzando una DTM come tabella di contingenza, attraverso la CA ogni testo diventa un vettore riga ed ogni token un vettore colonna nello spazio creato dagli assi ortogonali mantenuti. Questa rappresentazione permette di esplorare e valutare relazioni nella collezione di testi.

2.3.3 Trasformatori ed embedding

La rappresentazione delle parole tramite *embedding* è uno dei metodi più utilizzati per riprodurre il vocabolario di un testo. L'idea che ogni parola del linguaggio umano possa essere rappresentata da vettori numerici risale alla metà degli anni '50 (Firth, 1957), ma solo di recente si sono visti cambiamenti sostanziali nella letteratura, principalmente dovuti all'introduzione della struttura del trasformatore (Vaswani *et al.*, 2017). Con il termine *word embedding* si intende una lista di valori (un vettore ordinato) che mira a rappresentare numericamente il significato della parola (Kjell, 2023). Un *embedding*

generalmente comprende diverse centinaia di valori in modo che un singolo termine possa essere rappresentato in molte dimensioni. Di conseguenza, i valori presenti nel vettore possono essere visti come coordinate in uno spazio geometrico ad elevata dimensionalità. Più due parole sono vicine in questo spazio (per cui, più simili sono i vettori relativi alle due rappresentazioni), più ci si aspetta che le parole abbiano un significato simile. In altri termini, gli *embedding* catturano le relazioni tra le parole, dove la prossimità nello spazio multidimensionale indica una similarità nel significato. La rappresentazione di frasi e testi avviene attraverso l'aggregazione degli *embedding* relativi ai singoli token. Per fare ciò si può procedere adoperando la media, il massimo o il minimo di ogni dimensione dei singoli *embedding*.

L'obiettivo della creazione di uno spazio multidimensionale è registrare una qualche forma di relazione in esso, sia essa basata sul significato, sulla morfologia, sul contesto o su qualsiasi altro tipo di relazione. Poiché il linguaggio prodotto da autori diversi presenta differenze sistematiche a tutti i livelli della sua organizzazione linguistica (Mikros e Perifanos, 2013), è altamente probabile che rappresentazioni di questo tipo possano essere efficaci nel discriminare la scrittura di un modello linguistico da quella di un umano.

Produrre *embedding* di alta qualità necessita di un numero elevato di dati. Le parole all'interno del linguaggio naturale non si distribuiscono casualmente; al contrario, le parole di contesto sono prevedibili e definiscono il suo significato. Pertanto, è possibile utilizzare questa distribuzione per rappresentare "il significato di una parola attraverso i contesti in cui è stata osservata in un corpus" (Erk, 2012).

Inizialmente in letteratura erano sorti alcuni approcci che non tenevano conto di questi aspetti, consentendo di produrre solo *embedding* decontestualizzati; un esempio è *word2vec* (Mikolov *et al.*, 2013). In quei casi, le parole venivano trattate secondo un approccio "*bag of words*" che non considerava l'ordine ed il contesto in cui erano inserite. In seguito, si è tentato di perfezionare questi metodi nella capacità di identificare contenuti simili, utilizzando matrici di co-occorrenze per dare meno peso alle parole altamente frequenti. Algoritmi di questo tipo come *GloVe* (Pennington *et al.*, 2014), e *FastText* (Bojanowski *et al.*, 2017, Inc Facebook, 2016) consentono di identificare anche relazioni non lineari tra i termini. Approcci basati su tecniche di NLP più recenti hanno permesso di estrarre i significati latenti di un testo. Questi algoritmi si basano su strutture di *deep learning* capaci di produrre *embedding* contestualizzati, avendo come *input embedding* decontestualizzati per ogni parola, in grado di influenzarsi tra loro attraverso un meccanismo di *auto-attenzione*. Come risultato, la rappresentazione nello spazio multidimensionale relativa a un termine è diversa a seconda del contesto in cui è inserita e

dell'ordine con cui appare in una frase.

A questo scopo è possibile utilizzare i vettori costruiti da un modello linguistico pre-allenato in grado generare, in modo dinamico, rappresentazioni di termini che utilizzano l'informazione rappresentata dalle parole circostanti. Uno dei metodi più famosi per le rappresentazioni linguistiche, è BERT (Devlin *et al.*, 2019). Come discusso nel Capitolo 1, gli LLM sono composti da diversi strati, i cui *output*, chiamati stati nascosti, possono essere aggregati (ad esempio attraverso il concatenamento) per formare la rappresentazione tramite *embedding* di un token. Analogamente, gli *embedding* relativi a diversi token possono essere aggregati (ad esempio con la media) per rappresentare un testo. L'utilizzo di strati multipli permette di catturare relazioni non lineari. Non è ancora ben chiaro in che modo le informazioni differiscano nei diversi strati ma è stato dimostrato (Ethayarajh, 2019) come gli ultimi *layer* siano più specifici per il contesto mentre gli strati intermedi contengano una ricca gerarchia di informazioni linguistiche.

Durante l'addestramento dei modelli, il meccanismo di *auto-attenzione* permette di combinare le informazioni relative alle parole circostanti in modo che ognuna influenzi l'*embedding* della parola di riferimento. L'*auto-attenzione* pone pesi diversi agli *embedding* a seconda della rilevanza del termine a cui si riferiscono. Nel corso del processo, questi modelli sono esposti a una vasta gamma di testi e compiti linguistici, come il riempimento di spazi vuoti, la classificazione e la generazione di testi, che consentono loro di apprendere rappresentazioni linguistiche utili e generalizzabili. In questa fase, i modelli apprendono come parametrizzare gli strati di attenzione per amplificare l'influenza delle parti più rilevanti nel contesto. Una volta addestrati, i pesi della rete neurale contengono informazioni sulle relazioni semantiche tra le parole nel corpus di addestramento. Questi pesi possono essere utilizzati per creare *embedding* per nuove parole o nuovi testi, applicando la rete neurale ad essi.

Per ottenere *embedding* contestualizzati è possibile utilizzare il pacchetto `text` (Kjell, 2023) di `R` connesso alla libreria `transformers` (Wolf, 2019) di *Hugging Face* disponibile all'interno del linguaggio di programmazione `Python`. Il pacchetto permette di utilizzare vari modelli linguistici pre-addestrati in grado di produrre, su un testo di riferimento, *embedding* di alta qualità. I modelli si differenziano per le diverse lingue su cui è possibile applicarli e sul tipo di dati usati durante l'addestramento. Inoltre, possono disporre di un numero diverso di strati e stati nascosti. Questi aspetti vanno considerati durante la scelta del LLM più appropriato. I risultati migliori si ottengono quando è presente un'elevata similarità tra i materiali utilizzati per l'addestramento e i testi da analizzare, anche se può essere difficoltoso data l'enorme mole di dati adoperati nel *pre-training*. Anche il modo in cui vengono creati i token (definiti in questo caso come stringhe di

caratteri con un significato) può influire nelle prestazioni; comunemente ogni token riceve il suo *embedding*, ma le parole sconosciute al modello vengono divise in parti più piccole creando più rappresentazioni. Pertanto, la selezione del modello più appropriato e del numero di strati deve basarsi necessariamente su una ricerca empirica e ripetuta.

Alcuni modelli pre-addestrati, per testi in italiano, disponibili nella piattaforma sono:

- *bert-base-italian-uncased*: il modello è stato addestrato su un corpus di dimensione pari a 13GB (2 050 057 573 token) con informazioni provenienti principalmente dalla versione italiana di *Wikipedia* e dalla collezione di testi OPUS. Gli *embedding* prodotti sono rappresentati in 768 dimensioni
- *bert-base-multilingual-uncased*: il modello è stato addestrato su 102 lingue differenti utilizzando *Wikipedia* come principale fonte di dati, usando la tecnica del *Masked language modeling*(MLM)[3]. Gli *embedding* prodotti sono rappresentati in 768 dimensioni

La principale limitazione di questo potente approccio è che non è possibile mostrare l'importanza di singole parole o n-grammi nei modelli predittivi.

2.4 Analisi esplorative

In ambito stilometrico, le analisi esplorative sono fondamentali per comprendere le caratteristiche distintive dei testi e per investigare le somiglianze e differenze stilistiche tra diversi gruppi di documenti. Tra le tecniche più utilizzate per queste analisi figurano l'Analisi delle Componenti Principali (PCA; e.g. Wold *et al.*, 1987), il *Multidimensional Scaling* (MDS; e.g. Borg e Groenen 2005), la *Cluster Analysis*, e il *Bootstrap Consensus Tree*. Ognuna di queste tecniche, applicata all'intero corpus di 600 testi, offre un approccio unico per visualizzare e interpretare i dati.

In questi metodi, l'analisi si basa spesso sulle parole più frequenti (MFW). Come già sottolineato, l'utilizzo delle MFW è motivato dalla loro tendenza a essere rappresentative dello stile individuale e a potersi differenziare significativamente tra autori diversi. Analizzando queste parole, è possibile catturare le caratteristiche stilistiche che distinguono un gruppo di testi da un altro. In aggiunta, è anche comune utilizzare come base di analisi gli n-grammi e i char-grammi di grandezze diverse, oppure l'insieme di indicatori e quantità[4] relative ai testi, descritte nella Tabella 2.6. Queste misure possono arricchire

[3]Prendendo una frase, il modello maschera casualmente il 15% delle parole dell'*input* e, analizzando l'intera frase, il modello deve prevedere le parole mascherate.

[4]Nel seguito del capitolo verranno chiamate Text Features.

notevolmente la comprensione della struttura e dello stile dei testi e possono condurre a risultati più approfonditi e informativi.

L'obiettivo di queste analisi è di visualizzare come i testi si dispongono graficamente nello spazio delle componenti principali, dei *cluster* o della tecnica scelta, e di verificare se esista una netta separazione tra i documenti scritti dagli studenti e quelli prodotti dall'intelligenza artificiale. Una separazione netta suggerirebbe differenze stilistiche significative tra i gruppi, mentre una sovrapposizione indicherebbe somiglianze stilistiche.

In sintesi, le analisi offrono potenti strumenti per comprendere e visualizzare le caratteristiche stilistiche dei testi, facilitando la discriminazione a livello grafico tra gli autori, soprattutto quando vengono confrontati i risultati derivanti da tecniche differenti. Questi metodi consentono di ottenere una visione grafica della distribuzione dei testi e di valutare se sia possibile individuare una separazione netta tra di essi, considerando diverse rappresentazioni testuali. Per la tipologia e la natura delle tecniche seguenti è stata presa ispirazione dai contributi in letteratura del Professor M. Eder (2017, 2011) e dalle risorse disponibili nella libreria `Stylo` di `R` (Eder *et al.*, 2016b, 2016a).

2.4.1 Analisi delle Componenti Principali

L'Analisi delle Componenti Principali è una tecnica di semplificazione dei dati ampiamente usata nell'ambito della statistica multivariata (Shravan Kumar e Ravi, 2017). La PCA risponde all'esigenza di rappresentare un fenomeno k-dimensionale tramite un numero inferiore o uguale a k di variabili incorrelate, ottenute trasformando le variabili osservate. Questa tecnica permette di individuare delle combinazioni lineari delle variabili osservate che siano tra loro non correlate ed abbiano varianza massima, in modo da non disperdere informazioni. Applicando la PCA a matrici come le DFM, è possibile proiettare i testi in un nuovo sistema cartesiano, in cui la nuova variabile con la maggiore varianza viene proiettata sul primo asse, e la seconda nuova variabile per dimensione della varianza, sul secondo asse. Questa rappresentazione può essere utile a valutare se i testi appartenenti allo stesso autore si collocano nelle stesse parti del piano, rendendoli facilmente distinguibili.

In seguito, tramite la Figura 2.9 sono mostrati i risultati della proiezione sulle prime due componenti principali della DFM contente le frequenze relative, per ogni testo, delle parole più frequenti nel corpus. Sono stati utilizzati insiemi di MFW di grandezza crescente per verificare la consistenza dei risultati e identificare quante parole sono necessarie per differenziare sufficientemente i testi. La PCA è stata applicata sulla matrice di correlazione tra i dati.

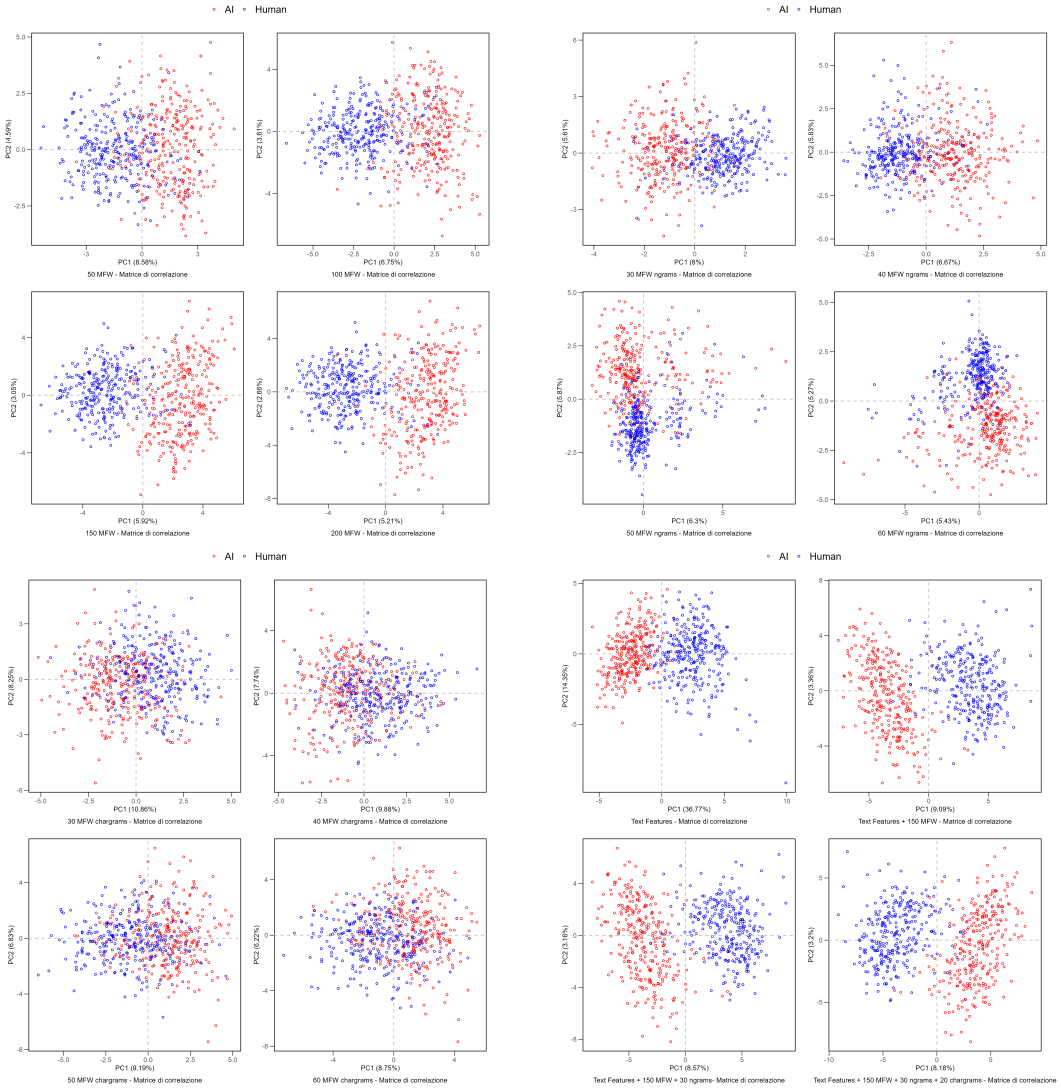


Figura 2.9: Grafico delle prime due componenti della PCA applicata a diverse DFM al variare del numero di MFW (in alto a sinistra), di n-grammi (in alto a destra) e di char-grammi (in basso a sinistra), e utilizzando varie combinazioni di *features* (in basso a destra).

Dai grafici relativi alle MFW, risulta che la distinzione migliore tra i testi si ottiene utilizzando le prime 150 o 200 parole più frequenti. Di conseguenza, i primi 100 token non sembrano sufficientemente rappresentativi per differenziare i testi e identificare i loro *pattern* distintivi. Al contrario, le prime due componenti principali mantengono una struttura simile tra i grafici con 150 e 200 MFW, indicando che l'aggiunta di

altre cinquanta parole non altera significativamente la distribuzione dei testi in questo spazio. Sebbene la distinzione non sia netta, la separazione tra i due gruppi rimane chiara e consistente nella terza specificazione (150 MFW), a dimostrazione del fatto che le informazioni stilistiche principali sono state catturate. Inoltre, la varianza spiegata dalle prime due componenti è comparabile e non varia particolarmente tra le diverse configurazioni, confermando la stabilità dei risultati.

Passando all'analisi degli n-grammi più frequenti, i risultati risultano essere peggiori. Le distinzioni tra le due categorie sono meno nette rispetto al caso precedente e il confine tra i due gruppi si dimostra più confuso. Gli n-grammi, se utilizzati singolarmente, non sembrano fornire particolari informazioni utili per distinguere i testi dal momento che la separazione grafica di questi ultimi è poco evidente.

La situazione peggiora ulteriormente utilizzando le sequenze di caratteri più frequenti. Tutti i grafici relativi ai char-grammi mostrano gruppi di punti sovrapposti e difficilmente distinguibili, specialmente in un intorno dell'origine degli assi.

Al contrario, creando una DMT basata sulle Text Features, e applicando la PCA, la distinzione tra i testi sulle prime due componenti principali risulta piuttosto netta. Risultati ancora migliori si ottengono combinando le Text Features con le 150 MFW, che portano a una separazione dei testi in due gruppi ben distinguibili nel piano. L'aggiunta di n-grammi e char-grammi non sembra essere particolarmente rilevante poiché non apporta nuove informazioni significative alla classificazione.

Queste analisi suggeriscono che, per future analisi, potrebbe essere più utile concentrarsi sull'utilizzo delle *feature* di testo e delle MFW piuttosto che sull'inclusione di n-grammi e char-grammi che sembrano essere già sufficientemente rappresentati.

2.4.2 Multidimensional Scaling

Il *Multidimensional Scaling* è una tecnica che permette di visualizzare la similarità o dissimilarità dei dati in uno spazio geometrico a bassa dimensione. In ambito stilometrico, il MDS viene utilizzato per rappresentare graficamente le distanze tra testi basate su determinate caratteristiche linguistiche o stilistiche. Questa tecnica consente di proiettare i testi in uno spazio bidimensionale o tridimensionale, dove la distanza tra i punti rappresenta la somiglianza stilistica: testi vicini nello spazio MDS sono stilisticamente simili, mentre testi lontani sono stilisticamente diversi. Per ottenere ciò si calcola una matrice di dissimilarità (o distanza) basata su varie misure (Euclidea, Manhattan, ...) a partire da una DFM. Questa tecnica riduce le dimensioni della matrice, preservando nel modo migliore possibile le distanze originali.

In seguito, sono mostrati i risultati dell'applicazione del MDS a DFM di varia natura, utilizzando le stesse quantità adottate nella PCA. Da alcuni tentativi in cui sono state adoperate tipologie di distanze diverse, è emerso che quella che fornisce i risultati migliori è la distanza *Delta*. Introdotta da J. Burrows (2002) è una misura che si basa sulla standardizzazione delle frequenze, dimostratasi spesso utile in questi contesti (Stanikūnas *et al.*, 2017). La distanza tra due testi è definita nel modo seguente:

$$\Delta_{(AB)} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{A_i - \mu_i}{\sigma_i} - \frac{B_i - \mu_i}{\sigma_i} \right|$$

dove n è il numero di *feature* o quantità simili utilizzate, A_i è la frequenza della *feature* i nel testo A, che viene confrontata con B_i che indica la frequenza della *feature* i nel testo B. La media dei valori delle frequenze della *feature* i nel corpus, e la loro deviazione standard, sono indicate rispettivamente con μ_i e con σ_i. Questa misura, semplice e di facile interpretazione, non dipende dal contenuto del testo ma dallo stile, rendendola robusta contro variazioni tematiche.

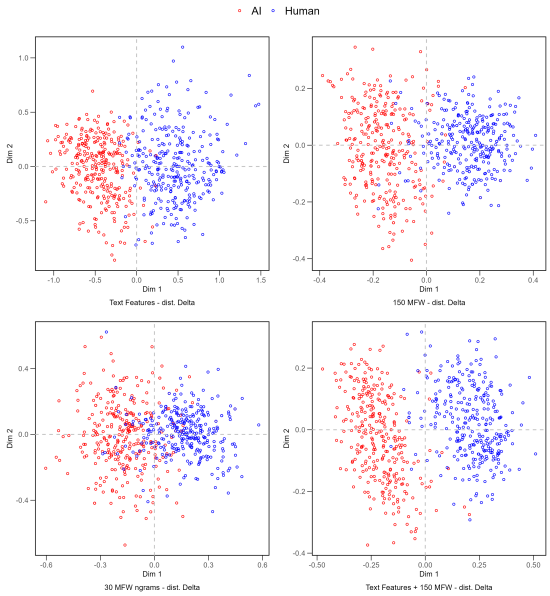


Figura 2.10: Grafico delle prime due dimensioni del MDS applicato a diverse DFM utilizzando varie combinazioni di *feature*.

I grafici in Figura 2.10 confermano i risultati ottenuti con la PCA. La separazione migliore si ottiene combinando le informazioni delle Text Features e le 150 MFW. La simile interpretazione è giustificabile tenendo conto del fatto che PCA e MDS sono tecniche somiglianti, entrambe che mirano a ridurre la dimensione dei dati preservando la

loro varianza. Analizzando i grafici nel complesso, è possibile affermare che si verifica più frequentemente la presenza di un testo Human nel gruppo AI, piuttosto che il contrario.

Seppur superficialmente, queste prime analisi esplorative suggeriscono che i testi prodotti da ChatGPT si identifichino abbastanza facilmente in un unico gruppo, mentre alcuni scritti umani potrebbero avere caratteristiche stilistiche che si avvicinano a quelle generate dall'AI; per cui è più probabile che vengano classificati in modo errato (Falsi Positivi).

2.4.3 Cluster Analysis

La *Cluster Analysis* è una tecnica di analisi multivariata adoperata per raggruppare osservazioni simili in insiemi distinti. Nell'ambito della classificazione dei testi, questa metodologia permette di identificare graficamente gruppi di testi con caratteristiche somiglianti, facilitando l'analisi delle differenze e delle similitudini tra vari documenti.

Per il suo utilizzo, si parte dalla creazione di alcune DFM, in modo analogo a quanto fatto nelle tecniche precedenti. Successivamente, si calcolano le distanze tra i testi applicate alle matrici, tramite la misura *Delta*, dal momento che si è dimostrata la più efficace per questo compito. Una volta ottenuta la matrice delle distanze, si applica il *clustering* gerarchico, utilizzando il metodo di *Ward* (Ward, 1963, Murtagh e Legendre, 2014). Questo metodo decompone la varianza per minimizzare la sua porzione all'interno dei *cluster*, assicurando che i testi raggruppati insieme siano il più omogenei possibile tra loro, ed eterogenei tra gruppi differenti. Il risultato di questa procedura è un dendrogramma: una rappresentazione grafica che mostra la gerarchia dei *cluster* formati. Tagliando il dendrogramma in due, è possibile identificare chiaramente due gruppi di testi e verificare la bontà del raggruppamento.

Tabella 2.8: Risultati di classificazione per diverse combinazioni di *feature*, metodi e distanze. Vengono mostrate solo le combinazioni che hanno ottenuti i valori più elevati in termini di Accuratezza (testi raggruppati correttamente).

Feature	**Distanza**	**Metodo**	**Accuratezza**	**Falsi Positivi**	**Falsi Negativi**
Text Features	*Delta*	`ward.D2`	576	16	8
	Delta	`ward.D`	574	20	6
150 MFW	*Delta*	`ward.D2`	551	28	21
	Delta	`ward.D`	553	35	12
150 MFW n-grammi	*Delta*	`ward.D2`	504	49	47
	Delta	`ward.D`	497	43	60
Text Features + 150 MFW	*Delta*	`ward.D2`	566	22	12
	Delta	`ward.D`	560	26	14

Creando DFM a partire dalle Text Features, dalle 150 MFW e dai 30 MFW n-grammi sono stati esaminati vari metodi di agglomerazione come "legame singolo", "legame com-

pleto", "legame medio" e altri, riscontrando che il metodo di *Ward* (`ward.D2`), e una sua specificazione alternativa (`ward.D`), producono i risultati migliori.

La Tabella 2.8 mostra che, in modo coerente alle tecniche precedenti, i raggruppamenti più precisi si ottengono utilizzando le Text Features e le prime MFW, mentre gli n-grammi non procurano ulteriori informazioni utili. Tuttavia, l'unione di Text Features e 150 MFW non fornisce miglioramenti rispetto all'analisi di *clustering* su queste quantità prese singolarmente, contrariamente a quanto avveniva con PCA e MDS. In quasi tutti i casi analizzati il numero di Falsi Positivi è superiore al numero di Falsi Negativi, che conferma le ipotesi stabilite osservando i grafici prodotti dal MDS.

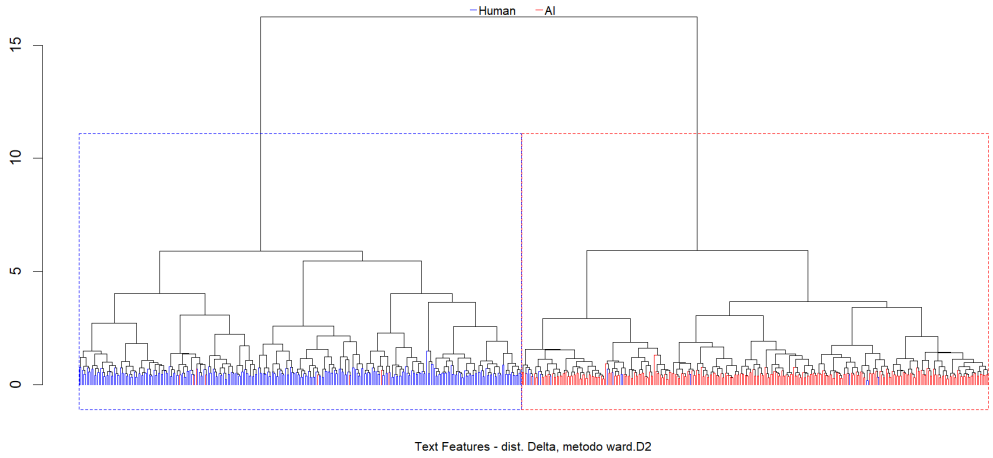


Figura 2.11: Dendrogramma della *Cluster Analysis* applicata ai testi utilizzando la distanza *Delta* e il metodo di `Ward.D2`.

Il dendrogramma in Figura 2.11 mostra tramite la sua struttura ad albero, la disposizione delle unità nei *cluster*, utilizzando la combinazione di parametri che per il *clustering* fornisce i risultati migliori, classificando correttamente 576 tesi su 600. Tagliando il dendrogramma in due, è possibile osservare la disposizione delle foglie nei *cluster* creati, e verificare la presenza di osservazioni posizionate nel gruppo errato (Falsi Positivi e Falsi Negativi).

2.4.4 Bootstrap Consensus Tree

La maggior parte dei metodi non supervisionati utilizzati nella stilometria presenta una limitazione significativa: è difficile valutare la robustezza del modello stimato. Questo

è il caso dell'Analisi delle Componenti Principali, del *Multidimensional Scaling* e della *Cluster Analysis*, di cui si è già discusso. Tuttavia, i risultati ottenuti con queste tecniche sono immediatamente interpretabili: permettono di individuare visivamente peculiarità o anomalie nel corpus analizzato. Inoltre, rappresentazioni grafiche come i dendrogrammi facilitano l'esposizione dei risultati in termini di raggruppamento dei testi. Inoltre, uno dei problemi principali della *Cluster Analysis* è la sua sensibilità al numero di *feature* analizzate (come le MFW), alle metriche di distanza e ai metodi di agglomerazione applicati. Stabilire quale combinazione di parametri offra la migliore separazione tra le unità statistiche non è semplice e può portare a scelte non ponderate o alla sola selezione dei risultati che confermano ipotesi preesistenti.

Una soluzione parziale a questo problema consiste nella combinazione delle informazioni ricavate da numerosi dendrogrammi in un unico grafico, noto come *Bootstrap Consensus Tree*. Questa tecnica, inizialmente sviluppata in filogenetica (Paradis *et al.*, 2004), è stata successivamente applicata in ambito linguistico (Dunn *et al.*, 2005) e stilometrico (Eder, 2013). L'idea è che, attraverso un gran numero di ripetizioni (ad esempio utilizzando un numero crescente di MFW), i raggruppamenti significativi tendano a riapparire frequentemente, mentre le somiglianze accidentali si manifestino solo sporadicamente. L'obiettivo è quindi ottenere modelli robusti tramite algoritmi di *clustering* ripetuti. Questa procedura genera una serie di dendrogrammi temporanei, dai quali si ricava un dendrogramma finale che mostra raggruppamenti più affidabili.

Il *Bootstrap Consensus Tree* in Figura 2.12 è stato ottenuto a partire dalle DFM relative alle parole più frequenti nel corpus, selezionandone inizialmente 100 e incrementando il loro numero di una unità fino ad arrivare a 200. Per ogni DFM è stata calcolata una matrice di dissimilarità tramite la distanza *Delta*, a cui successivamente è stato applicato il metodo di collegamento `ward.D2` per ottenere un largo numero di alberi di classificazione, leggermente diversi tra loro. Per sintetizzare le informazioni provenienti dai diversi dendrogrammi è stata calcolata la frequenza con cui ogni raggruppamento appare in ciascun albero e sono stati inclusi nel *Bootstrap Consensus Tree* solo i raggruppamenti dei testi presenti in almeno il 50% dei casi. Associare un testo ad un determinato autore solo se quest'ultimo gli viene attribuito più della metà delle volte, genera gruppi stabili e affidabili, consentendo di ottenere una rappresentazione più veritiera e robusta della struttura del corpus. Tuttavia, nemmeno in questo caso si ottiene una separazione ottimale, sebbene la quasi totalità degli scritti venga classificata correttamente.

Le diverse analisi esplorative effettuate sul corpus hanno permesso di esaminare la sua struttura e le relazioni tra i testi in modo efficace e comprensibile. Tramite le tecniche statistiche adoperate è stato possibile visualizzare le prime somiglianze e differenze

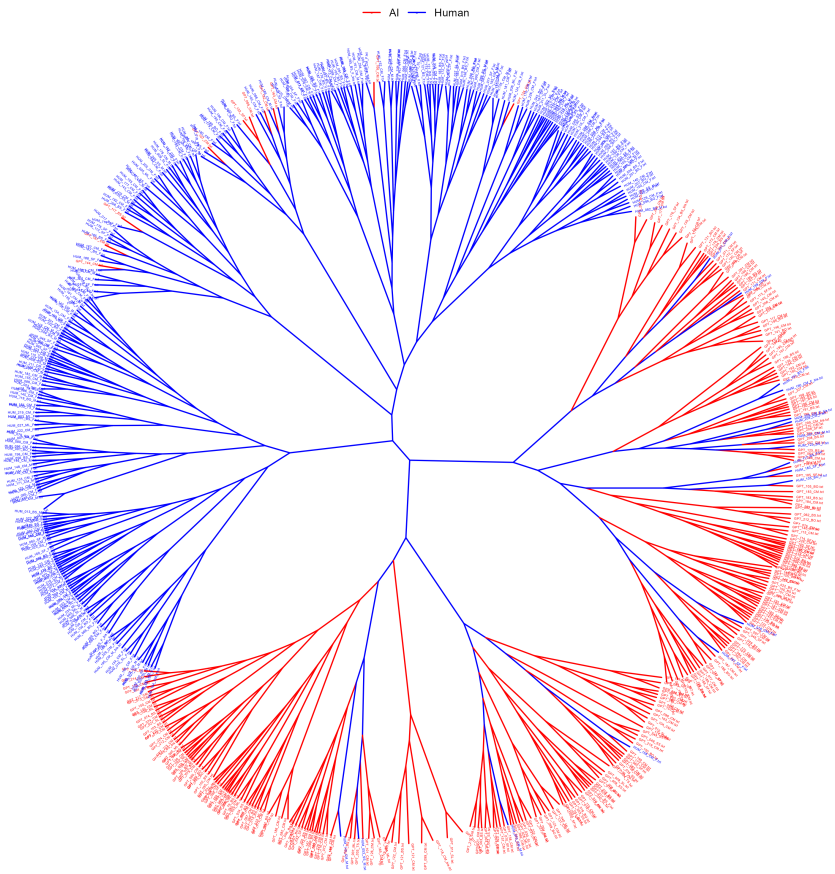


Figura 2.12: *Bootstrap Consensus Tree* della classificazione dei testi.

tra i vari documenti, identificando spessi i due gruppi in modo distinto, senza però mai raggiungere un risultato ideale. Per procedere verso una classificazione più rigorosa e affidabile, è necessario passare a modelli statistici avanzati e metodologie più complesse che permetteranno di ottenere risultati più precisi. I risultati preliminari delle analisi esplorative potranno essere confermati e approfonditi, per raggiungere lo scopo di assegnare in modo automatico l'intero corpus di testi al relativo autore.

Capitolo 3

Il modello di classificazione

Nel capitolo precedente è stata descritta la costruzione del corpus di testi e il trattamento iniziale dei dati tramite il *lexical-processing*, in modo da renderli utilizzabili per le analisi successive. Dopo aver esplorato le proprietà fondamentali della raccolta e le principali relazioni tra i testi, sono stati introdotti tre diversi metodi di vettorizzazione: la *feature extraction*, l'uso di trasformatori ed *embedding* e l'utilizzo di tecniche di riduzione della dimensionalità. Questi approcci sono essenziali per trasformare il testo grezzo in una rappresentazione numerica adatta ai modelli di *machine learning*.

In questo capitolo l'interesse verterà principalmente sulla fase di classificazione. Verranno discusse le quantità generate tramite i metodi di vettorizzazione e saranno descritti i modelli che verranno utilizzati per predire le categorie dei testi. In particolare, saranno valutate le loro prestazioni ed esplorate alcune tecniche di ottimizzazione per migliorare i risultati (*fine-tuning*, selezione degli iperparametri, convalida incrociata, ...). Attraverso un'analisi rigorosa e il preciso uso dei modelli di classificazione, l'obiettivo è ottenere previsioni accurate, in grado di associare ogni testo al suo autore col minore grado di errore possibile.

3.1 Costruzione delle previsioni

La procedura tipica dei problemi di classificazione prevede di partire dai dati ottenuti dal processo di vettorizzazione dei testi e organizzarli in strutture chiamate *dataframe*, che contengono sia le rappresentazioni numeriche dei testi, sia l'etichetta relativa all'autore associato ad essi. I dati vengono divisi in due porzioni di differente numerosità: la prima, più grande, relativa al *training set*, e la seconda relativa al *test set*. In questo caso si è deciso di applicare una suddivisione che prevede proporzioni pari a 80% e 20% del totale.

I dati contenuti nel *training set* sono adoperati per allenare dei modelli di classificazione che saranno poi utilizzati per prevedere la variabile risposta (cioè l'autore) dei testi contenuti nel *test set*. Il fine ultimo è valutare le prestazioni di vari modelli, utilizzando approcci differenti per la rappresentazione delle caratteristiche dei testi. Le *performance* sono stabilite, tramite alcune metriche, in base all'accuratezza nell'attribuire ad ogni unità statistica nel *test set* l'autore corretto.

3.1.1 Modelli di classificazione

I modelli di classificazione sono strumenti di *machine learning* che permettono di assegnare etichette o categorie ai dati di *input*, costituiti in questo caso dai testi o dalle loro rappresentazioni. La scelta del modello di classificazione dipende da vari fattori, tra cui la natura dei dati, la tipologia di classificazione (binaria o multi-classe), la dimensione del *dataset* e le specifiche esigenze di interpretabilità e velocità di esecuzione.

L'uso di molteplici modelli consente di confrontare e selezionare l'algoritmo che offre le migliori prestazioni per i dati in questione. Ogni modello ha i suoi punti di forza e di debolezza, e la scelta può influenzare significativamente la qualità delle previsioni. Alcuni modelli possono essere più efficaci nel catturare relazioni complesse e non lineari nei dati, mentre altri possono essere più robusti rispetto al rumore o ai dati anomali. Inoltre, alcuni modelli sono più interpretabili, il che può essere importante per comprendere come vengono effettuate le previsioni.

Dal numeroso elenco di scelte disponibili nella libreria `caret` (Kuhn, 2008) di `R`, sono stati selezionati i seguenti dieci modelli; si rinvia all'Appendice per una formalizzazione dei principali:

- `svmRadial` (*Support Vector Machine* con *kernel* radiale; Scholkopf *et al.*, 1997): utilizza un *kernel* radiale per gestire problemi non lineari, creando un iperpiano che separa le classi in uno spazio ad alta dimensione
- `svmRadialWeights` (*Support Vector Machine* con *kernel* radiale e pesi): una variante della SVM con *kernel* radiale che assegna pesi differenti alle classi, utile per gestire problemi di squilibrio nelle classi
- `svmLinear` (*Support Vector Machine* con *kernel* lineare; Vapnik *et al.*, 1998): utilizza un *kernel* lineare per trovare l'iperpiano ottimale che separa le classi, efficace per problemi di classificazione lineare
- `rf` (*Random Forest*; Brieman, 2001): un metodo che combina le previsioni di più modelli base creando una "foresta" di alberi decisionali, utilizzando il voto della maggioranza per le previsioni

- `Rborist` (*Random Forest* ottimizzato; Seligman, 2015): una versione ottimizzata di *Random Forest*, focalizzata su velocità di esecuzione ed efficienza di memoria
- `ranger` (*Random Forest*; Wright e Ziegler, 2017): un'implementazione efficiente di *Random Forest*, che utilizza più alberi decisionali per migliorare la precisione per dati ad elevata dimensione
- `cforest` (*Conditional Inference Random Forest*; Hothorn *et al.*, 2004): una variante di *Random Forest* che utilizza *test* condizionali per la creazione degli alberi, riducendo la distorsione da selezione delle variabili
- `xgbTree` (*eXtreme Gradient Boosting*; Chen e Guestrin, 2016): implementa il *boosting* di alberi decisionali, ottimizzando la funzione di perdita e migliorando le previsioni tramite combinazioni di modelli deboli
- `gbm` (*Stochastic Gradient Boosting*; Friedman, 2001): costruisce modelli incrementali, dove ogni nuovo modello corregge gli errori del modello precedente, ottimizzando la funzione di perdita
- `glmnet` (*Generalized Linear Model* con penalizzazione; Friedman *et al.*, 2010): combina la regressione lineare generalizzata con tecniche di penalizzazione *elastic net* (Lasso e Ridge) per migliorare la generalizzazione

Il *Random Forest* è un modello di *bagging* ampiamente utilizzato nel *text mining* per la sua robustezza e capacità di gestire dati anche ad alta dimensionalità, che spesso ottiene i risultati migliori in problemi di questo tipo (Pal, 2005). Sono state incluse alcune sue varianti ottimizzate per sfruttare i loro miglioramenti in termini di efficienza e accuratezza. I modelli di *boosting*, come `xgbTree` e `gbm`, sono noti per le loro eccellenti *performance* nel combinare modelli deboli per ottenere un modello più potente (Sutton, 2005). In particolare, il *boosting* per gli alberi è molto efficace nel migliorare la precisione delle previsioni. Le SVM sono comunemente utilizzate per problemi di classificazione grazie alla loro capacità di creare iperpiani ottimali per separare le classi. Le varianti inserite permettono di esplorare diverse funzioni *kernel* e approcci per gestire dati anche in modo non lineare. Infine, `glmnet` è utile per la selezione automatica delle variabili più rilevanti e per migliorare la generalizzazione, riducendo il rischio di *overfitting*. Questo modello combina la regressione lineare generalizzata con tecniche di penalizzazione, offrendo un equilibrio tra interpretabilità e *performance* predittive.

3.1.2 Metodi di Train Control

Il controllo dell'addestramento, o *train control*, è una componente fondamentale nella costruzione dei modelli di *machine learning*, poiché determina come verranno valuta-

ti e ottimizzati i modelli durante la fase di addestramento. L'obiettivo principale del *train control* è garantire che i modelli siano valutati in modo affidabile e generalizzabile, prevenendo problemi come l'*overfitting* e migliorando la capacità del modello di agire efficacemente su dati non conosciuti. Il pacchetto `caret` fornisce diverse opzioni per questo compito, che consentono di definire accuratamente i metodi di validazione e di ottimizzazione utilizzati durante l'addestramento. Dalle tipologie disponibili sono state selezionate cinque differenti specificazioni:

- `repeatedcv` (*Repeated Cross-Validation*): la convalida incrociata ripetuta suddivide il *training set* in un determinato numero di parti (*fold*), in questo caso 5, e addestra il modello su diverse combinazioni di questi *fold*, ripetendo il processo per un certo numero di volte (in questo caso 15). Questo metodo fornisce una stima più stabile e affidabile delle *performance* del modello, riducendo la varianza dei risultati rispetto alla semplice convalida incrociata
- `optimism_boot` (*Optimism Bootstrap*; Efron e Tibshirani, 1994): il *bootstrap* è un metodo che crea molteplici ricampionamenti con sostituzione del *dataset* originale. L'*optimism bootstrap* corregge il fattore di ottimismo dato dalle apparenti *performance* predittive, nella valutazione dei modelli. Questo metodo fornisce stime accurate e, utilizzando cento ricampionamenti, produce una valutazione robusta e attendibile dei risultati
- `adaptive_boot` (*Adaptive Bootstrap*): questo metodo che combina il *bootstrap* tradizionale (Efron e Tibshirani, 1994) con tecniche adattive per ottimizzare ulteriormente il processo di addestramento. Questo approccio permette di concentrarsi sulle parti più complicate del *training set*, migliorando la capacità del modello di generalizzare su casi complessi. Anche in questo caso, con 100 ricampionamenti, si garantisce una valutazione accurata delle *performance*
- `adaptive_cv` (*Adaptive Cross-Validation*): è una tecnica avanzata che adatta il processo di convalida incrociata in base alle prestazioni intermedie del modello. Questo metodo può adattare il numero di *fold* o la composizione dei dati di *training* e convalida durante l'addestramento. Questa tecnica permette di ottimizzare il processo di convalida in modo dinamico, concentrandosi su aree problematiche dei dati e migliorando l'accuratezza delle stime delle *performance*. È particolarmente utile per modelli complessi o *dataset* con numerose variabili
- `LGOCV` (*Leave-Group-Out Cross-Validation*): è una variante della convalida incrociata, che divide ripetutamente il *training set* in gruppi, lasciandone fuori uno o più in ogni iterazione. La maggior parte dei gruppi viene utilizzata per l'addestramen-

to e i restanti per la validazione. Questa tecnica è particolarmente utile quando si vuole assicurare che la convalida avvenga su segmenti specifici ed eterogenei dei dati

Le tecniche utilizzate rientrano in due categorie: convalida incrociata e *bootstrap*. L'idea alla base di questi approcci è simile: ripetendo l'adattamento più volte su diversi sottoinsiemi di dati di addestramento, è possibile comprendere meglio l'entità dell'*overfitting* e tenerne conto nella costruzione e nella valutazione dei modelli. Nei metodi adattivi non viene eseguito l'intero *set* di ricampionamento per ciascun modello; al contrario, durante questo processo, viene condotta un'analisi di rilevanza e i modelli con una bassa probabilità di essere ottimali vengono rimossi (Kuhn, 2014). Inoltre, per migliorare l'efficienza e ridurre i tempi computazionali del processo di addestramento e validazione dei modelli, è stato utilizzato il calcolo in parallelo. Distribuendo il carico di lavoro su più processori (*core*) è possibile eseguire simultaneamente più operazioni, velocizzando i calcoli richiesti e sfruttando a pieno le risorse del processore.

3.1.3 Dati di addestramento e di verifica

I metodi di vettorizzazione descritti nel capitolo precedente permettono di costruire di *dataset* su cui, dopo la suddivisione in *training* e *test set*, verranno applicati i modelli di classificazione per la previsione dell'autore per ogni testo. Per ogni tipologia, sono stati creati *dataframe* di dimensioni diverse variando il numero di caratteristiche (colonne) incluse. Questo approccio permette di analizzare se e come la quantità di informazioni influisce sui risultati dei modelli. In altre parole, valutare se un numero maggiore o minore di predittori influisce positivamente o negativamente sulla capacità dei modelli di classificare correttamente i testi. Attraverso questa analisi, sarà possibile identificare il bilanciamento ottimale tra la dimensionalità del *dataset* e le *performance* dei modelli, fornendo indicazioni utili per future applicazioni di *text mining* e classificazione.

Feature Extraction

Per la prima tipologia di vettorizzazione denominata *feature extraction* sono state utilizzate le Text Features descritte nel Capitolo 2, le MFW, i MFW *n-grams*, i MFW *char-grams* e alcune combinazioni di queste quantità. Servendosi di *dataframe* con composizioni differenti è possibile valutare la loro capacità di addestrare i modelli in modo efficace e determinare se unire informazioni di tipologie diverse porta a risultati migliori. Nel dettaglio, sono state utilizzate le seguenti configurazioni:

- Text Features: una selezione di 25 indicatori linguistici ricavati dalle Tabelle 2.1 e 2.6
- MFW: una selezione dei 150 token più frequenti nel corpus
- n-grams: i 30 n-grammi più frequenti nel corpus
- char-grams: i 30 char-grammi più frequenti nel corpus
- Text Features + MFW: l'unione di Text Features e MFW
- Text Features + MFW + n-grams: l'unione di Text Features, MFW e n-grammi

A partire dalle singole quantità sono stati creati *dataframe* composti da 600 righe, una per ogni testo del corpus, e da un numero di colonne pari alle caratteristiche che si vogliono utilizzare. Nel primo caso sono stati selezionati gli indicatori che, in seguito alle analisi esplorative, sembravano contribuire maggiormente alla discriminazione dei testi. Nel secondo caso, a partire dalla DFM relativa all'intero corpus sono stati estratti i token con le frequenze assolute più elevate.

Tuttavia, per garantire che i risultati siano applicabili in vari contesti, è fondamentale utilizzare parole che non rivelino l'argomento specifico di un testo. A questo scopo, conviene concentrarsi su parole vuote (*stopwords*) o parole molto generiche. Utilizzando queste tipologie di token, si evita di far emergere il tema trattato nei testi, garantendo che l'analisi stilometrica si basi esclusivamente sugli aspetti stilistici piuttosto che sul contenuto. A questo proposito sono stati rimossi i token ritenuti troppo specifici e che difficilmente potrebbero essere frequenti in contesti più generici, fino ad ottenere una selezione di 150 parole come preposizioni, aggettivi, avverbi, pronomi e sostantivi generici. Nel caso degli n-grammi sono stati selezionati i 30 con le frequenze assolute più elevate, senza alcuna particolare selezione. La stessa cosa è avvenuta per i char-grammi.

Ad eccezione del caso delle Text Features, i *dataframe* sono stati convertiti passando alle frequenze relative. Questa trasformazione è utile per la normalizzazione dei dati che rende comparabili testi di lunghezze diverse. Senza questa correzione i documenti più lunghi tenderebbero ad avere conteggi assoluti più alti semplicemente a causa della loro lunghezza, introducendo una distorsione nei modelli di classificazione. Dal momento però che i testi, per costruzione, hanno lunghezze simili, questo aspetto diventa poco rilevante. Esiste però un'altra ragione per cui conviene passare alle frequenze relative: il loro utilizzo riduce la varianza introdotta da documenti con conteggi di parole molto diversi. Ciò contribuisce a migliorare la stabilità e l'affidabilità dei modelli. Le ultime due tipologie di *dataframe* hanno lo scopo di analizzare se la combinazione di *feature* differenti contribuisce a migliorare i risultati, nel caso non si raggiungesse una classificazione ottimale con le quantità singole. Inoltre, in un'ottica di selezione delle variabili e di

analisi della loro importanza, permette di evidenziare le caratteristiche complessivamente più rilevanti per i testi.

Analisi delle Corrispondenze

Il secondo metodo di vettorizzazione dei testi riguarda la riduzione della dimensionalità tramite l'Analisi delle Corrispondenze. Partendo dalla DFM relativa al corpus si selezionano i token con un numero minimo di occorrenze pari 50 nell'intero insieme di testi. Questa operazione permette di concentrarsi su un numero di *feature* ridotto che, verosimilmente, contiene la maggior parte dell'informazione necessaria a descrivere i testi. Grazie a questa operazione il numero di colonne nella DFM passa da 15 761 a 693. Inoltre, in seguito ad alcune prove, è possibile affermare che le *performance* dei modelli non variano significativamente a seconda del numero di *feature* selezionate per la CA e che quindi è possibile procedere con questa configurazione. Sulla matrice risultante si applica l'Analisi delle Corrispondenze, che permette di proiettare i testi su un numero definito di assi con importanza decrescente.

Le righe della matrice relative ai documenti nel *test set* vengono trattate come righe supplementari. Le righe supplementari sono osservazioni che non vengono utilizzate per costruire la soluzione dell'Analisi delle Corrispondenze, ma vengono proiettate nello spazio delle Componenti Principali ottenuto mediante le righe che partecipano attivamente. Le unità di *test* vengono quindi visualizzate nello stesso spazio delle unità di *training*, senza influenzare la decomposizione originale dei dati. Questo passaggio assicura che il modello sia costruito esclusivamente sui dati di addestramento, mantenendo così l'indipendenza necessaria per una corretta stima dei risultati e rendendo il processo analogo alla validazione del modello in un contesto di *machine learning*. Inoltre, tramite questa accortezza, è possibile valutare come i dati di *test* si comportano rispetto alla struttura dei dati di addestramento. Ciò permette di verificare se i *pattern* identificati nel *training set* si replicano nel *test set*, e di individuare eventuali testi anomali o *outlier*.

La prima configurazione dei dati prevede di utilizzare tutti gli assi risultanti a disposizione. Per come è definita la CA, il numero massimo di colonne (assi) adoperabili è pari al numero di righe meno uno. Dal momento che il *training set* costituisce l'80% dei 600 testi, il numero massimo di assi su cui proiettare i testi è pari a 479. In analogia col metodo precedente, si costruiscono due configurazioni alternative ridotte che richiedono un numero inferiore di assi con cui descrivere i dati. La prima utilizza i primi 150 assi, mentre per la seconda conviene analizzare in modo più approfondito le quantità in analisi.

Visto che l'obiettivo è ridurre la dimensionalità, mantenendo al contempo la maggior parte dell'informazione presente, è opportuno calcolare l'inerzia degli assi e identificare il punto in cui essa inizia a diminuire notevolmente. L'inerzia è una misura della varianza spiegata da ciascun asse. Gli assi principali nell'Analisi delle Corrispondenze rappresentano direzioni nello spazio derivante dalla decomposizione dei dati di partenza, che spiegano la massima quantità di varianza. L'inerzia totale è una misura complessiva della varianza nel *dataset* originale, e ciascun asse contribuisce ad essa. Tipicamente, i primi pochi assi catturano la maggior parte dell'inerzia totale e man mano che si considerano assi successivi, l'inerzia spiegata da ciascun asse tende a diminuire.

Il "punto di gomito" è il punto nel quale la quantità di varianza spiegata da ulteriori assi comincia a diminuire drasticamente. In altre parole, è il punto in cui aggiungere ulteriori dimensioni non contribuisce significativamente alla spiegazione della varianza totale. Identificare questo punto permette di diminuire il costo computazionale, mantenendo la maggior parte dell'informazione per le analisi successive. In aggiunta, consente di ottenere risultati maggiormente interpretabili, focalizzandosi sulle dimensioni che realmente contano.

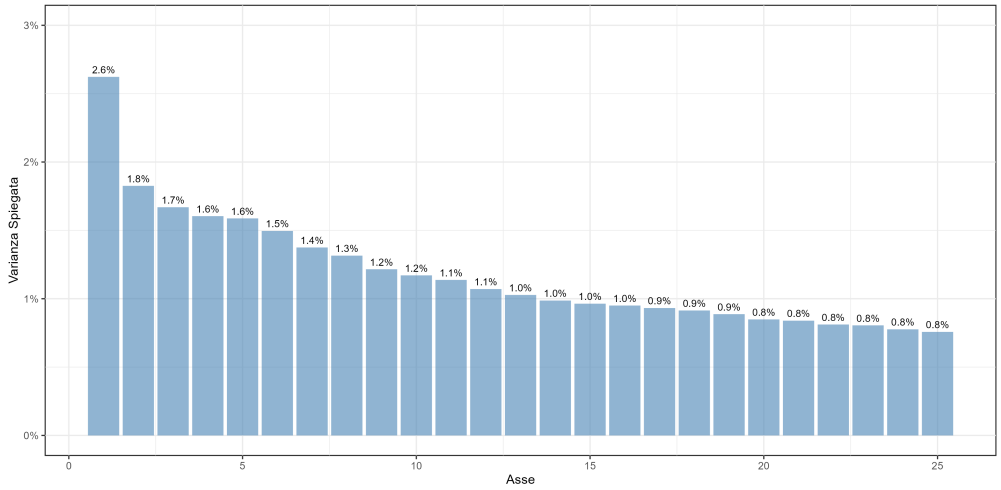


Figura 3.1: Grafico a barre raffigurante la percentuale di varianza spiegata dai primi 25 assi nell'Analisi delle Corrispondenze calcolata sui dati completi.

Dalla Figura 3.1 si può vedere come i valori di varianza spiegata decrescono in modo abbastanza lineare, rendendo difficile l'individuazione di un punto di gomito. Osservando però il contributo di ogni singolo asse si può decidere di limitarsi all'uso delle dimensioni che spiegano almeno l'1% della varianza totale. Per questi motivi, la seconda configurazione ridotta prevede l'utilizzo dei primi 15 assi derivanti dall'Analisi delle

Corrispondenze, in grado di spiegare circa il 21% della varianza totale. Per l'approccio mediante la CA si è dunque deciso di adoperare le seguenti quantità:

- CA-479: tutti i 479 assi derivanti dalla decomposizione iniziale
- CA-150: i primi 150 assi (con varianza spiegata pari al 76% del totale)
- CA-15: i primi 15 assi (con varianza spiegata pari al 21% del totale)

La scelta di utilizzare 150 assi consente di porre in competizione questa tipologia di rappresentazione numerica dei dati con la precedente, riducendo progressivamente le dimensioni e preservando al contempo la loro capacità predittiva ed esplicativa.

Large Language Models

Come descritto nel capitolo precedente, il terzo e ultimo metodo di vettorizzazione considerato prevede l'uso di strutture chiamate trasformatori, che consentono la raffigurazione dei testi tramite *embedding*. È possibile effettuare questo processo tramite la funzione `textEmbed` della libreria `Text` di `R`, che permette di operare ad alto livello producendo rappresentazioni contestualizzate. Dal momento che il corpus di testi è in lingua italiana, è necessario servirsi di modelli addestrati su materiali di questo tipo. Di conseguenza, si è deciso di utilizzare un LLM capace di operare su più lingue e uno più specifico per l'italiano, presumibilmente in grado di produrre risultati più accurati. La scelta dei modelli linguistici pre-allenati, tramite i quali creare gli *embedding*, è ricaduta su *bert-base-multilingual-uncased* e *bert-base-italian-uncased*.

Modelli diversi possono basarsi su tipologie differenti di materiali di addestramento e non separare i token allo stesso modo; questi aspetti vanno considerati in una successiva interpretazione dei risultati. Inoltre, i modelli selezionati convertono tutte le parole in formato minuscolo. Questa scelta è coerente con il *pre-processing* effettuato in precedenza e, riducendo il numero di *type* da considerare, ha più possibilità di portare alla creazione di *embedding* di maggiore qualità. Anche il numero di *layer* utilizzati svolge un ruolo importante all'interno del processo. Questa decisione dovrebbe basarsi su ricerche empiriche e sistematiche ma, per compiti come previsione dell'età o del genere dell'autore di un testo, è stato generalmente riscontrato (Matero *et al.*, 2019, Ganesan *et al.*, 2021) che l'utilizzo degli ultimi quattro strati o del penultimo strato produce buoni risultati.

Tuttavia, per mantenere una maggiore generalità dei risultati si è deciso di adoperare tutti i 12 *layer* a disposizione, ad eccezione del primo, relativo agli *embedding* di *input* al modello. Dal momento che questo strato non è contestualizzato, non va usato. La funzione `textEmbed` consente di gestire l'aggregazione da strati a token e da token a testi. Il primo passaggio viene fatto attraverso il concatenamento e il secondo attraverso la

media. Questi valori predefiniti seguono le convenzioni presenti in letteratura riguardanti l'utilizzo dei trasformatori per compiti di classificazione (Ganesan *et al.*, 2021). Questa configurazione dei parametri permette la creazione di *embedding* contestualizzati per ogni testo, ognuno proiettato su 768 dimensioni numeriche, come previsto dai modelli BERT.

La creazione delle rappresentazioni numeriche è avvenuta fornendo in *input* alla funzione l'intero corpus di testi, effettuando in un secondo momento la divisione dei vettori in *training set* e *test set*. Così facendo, non si sta influenzando la creazione delle stime utilizzando dati che dovrebbero essere sconosciuti al modello, dal momento che modelli come BERT sono già ottimizzati per comprendere le strutture linguistiche generali. LLM di questo tipo applicano un modello già formato a ciascun testo in modo indipendente: ogni riga del *dataset* (cioè, ogni testo) viene trasformata in un *embedding* in modo indipendente dagli altri testi. Di conseguenza, la generazione degli *embedding* non utilizza informazioni specifiche del *test set*, mantenendo l'integrità del processo di valutazione.

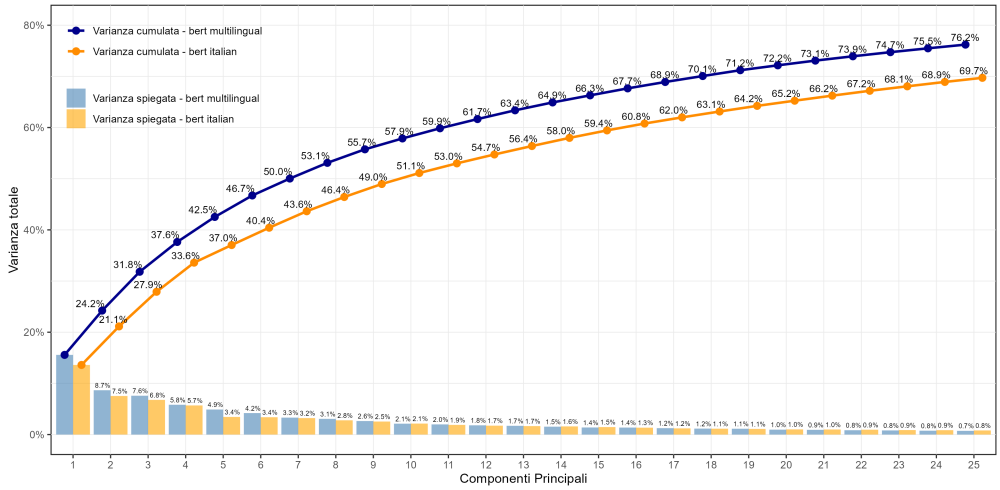


Figura 3.2: Grafico relativo ai valori di varianza spiegata dalle prime 25 Componenti Principali applicate vettori relativi al modello linguistico *bert-base-multilingual-uncased* in azzurro e al modello linguistico *bert-base-italian-uncased* in giallo. La linea spezzata blu indica l'incremento della varianza cumulata al variare del numero di componenti per il modello multilingua, mentre la linea spezzata arancione è relativa al modello esclusivamente italiano.

Anche per questa tipologia di raffigurazione, la prima coppia di *dataframe* in uso comprende tutte le 768 dimensioni (i vettori) disponibili. Successivamente, per ridurre la grandezza dei dati e creare quantità intermedie, è possibile utilizzare la tecnica della PCA per proiettare i vettori in un nuovo sistema cartesiano con un numero ridotto di

componenti, cercando di preservare al massimo l'informazione presenti nei dati. Per operare in linea con le analisi precedenti si è deciso inizialmente di creare, per ognuno dei due LLM, un *dataset* ristretto attraverso le prime 150 componenti principali. In aggiunta, per ridurre ulteriormente il numero di dimensioni si può procedere osservando l'andamento della varianza spiegata da ogni componente, cercando nuovamente il punto di gomito nel grafico.

Nel caso del modello linguistico *bert-base-multilingual-uncased* in Figura 3.2 non è ben chiaro a che livello si posizioni il punto di gomito, anche se già dopo i primi 20 valori la varianza spiegata si stabilizza. Si può notare invece come bastino solamente le prime 25 componenti principali per superare ampiamente il 75% di varianza cumulata. Per rendere efficace il confronto con il metodo dell'Analisi delle Corrispondenze, si decide di utilizzare i dati fino alla quindicesima componente principale, per creare la configurazione più ridotta per questa tipologia di raffigurazione dei testi. Anche per il secondo modello non è facile individuare un punto di gomito nell'andamento delle varianze. In questo caso i valori sono leggermente inferiori ma con le prime 25 componenti si riesce a spiegare comunque il 70% della variabilità dei dati. Nuovamente, si sceglie di utilizzare le prime 15 componenti principali per creare l'ultimo *dataframe* da mettere in competizione con i precedenti.

Di conseguenza, in seguito alle scelte effettuate le quantità che saranno poi utilizzate dai modelli di classificazione sono le seguenti:

- Bert-multi-768: si utilizzano tutti i vettori disponibili prodotti da *bert-base-multilingual-uncased*
- Bert-multi-150: si utilizzano le prime 150 componenti (varianza spiegata pari al 96.5%) dopo aver applicato la PCA sui vettori prodotti da *bert-base-multilingual-uncased*
- Bert-multi-15: si utilizzano le prime 15 componenti (varianza spiegata pari al 66.3%) dopo aver applicato la PCA sui vettori prodotti da *bert-base-multilingual-uncased*
- Bert-ita-768: si utilizzano tutti i vettori disponibili prodotti da *bert-base-italian-uncased*
- Bert-ita-150: si utilizzano le prime 150 componenti (varianza spiegata pari al 95.5%) dopo aver applicato la PCA sui vettori prodotti da *bert-base-italian-uncased*
- Bert-ita-15: si utilizzano le prime 15 componenti (varianza spiegata pari al 59.4%) dopo aver applicato la PCA sui vettori prodotti da *bert-base-italian-uncased*

L'analisi dei dati ha portato all'adozione di tre approcci equiparabili per la rappresentazione dei testi, ognuno dei quali è stato esaminato anche con versioni ridotte, per

bilanciare l'accuratezza dei risultati e il costo computazionale. La scelta di utilizzare tre tecniche differenti è motivata dalla volontà di ottenere una panoramica completa e accurata delle caratteristiche dei testi. Ogni metodo mette in luce aspetti distinti dei testi: mentre le *feature* lessicali offrono una visione diretta e quantificabile delle caratteristiche linguistiche, l'Analisi delle Corrispondenze rivela schemi latenti e i *Large Language Models* forniscono rappresentazioni semantiche profonde.

Confrontando le *performance* dei modelli di classificazione, è possibile determinare quale metodo di rappresentazione dei testi sia più efficace per il compito di previsione dell'autore. Sebbene le *feature* siano più facilmente interpretabili, l'Analisi delle Corrispondenze e i modelli linguistici offrono prospettive più astratte e semantiche, assicurando che i risultati siano sostenuti da una varietà di rappresentazioni e tecniche.

3.2 Confronto dei risultati

Dal momento che si sta affrontando un problema di classificazione binaria, i modelli vengono addestrati sui dati di *training* e, applicandoli successivamente al test set, si calcola la probabilità che ciascun testo sia stato scritto da uno dei due autori (in questo caso è stata posta GPT, relativa ai testi AI, come classe positiva di riferimento). Una probabilità maggiore o uguale a 0.5 indica che il testo è stato scritto da ChatGPT, mentre una probabilità minore di 0.5 indica che il testo è stato scritto da uno studente. Il confronto per ogni combinazione di tipologia di rappresentazione, modello linguistico e controllo dell'addestramento (*train control*) avviene basandosi sull'accuratezza con cui ogni testo nel *test set* viene attribuito al corretto autore. A questo scopo si confrontano i risultati mediante le seguenti quantità:

- *Accuracy*: misura la percentuale di previsioni corrette effettuate dal modello. Viene calcolata come il numero di previsioni veritiere (Veri Positivi + Veri Negativi) diviso il numero totale di previsioni (Veri Positivi + Veri Negativi + Falsi Positivi + Falsi Negativi)
- *Kappa*: il Kappa di Cohen è un indice di concordanza che tiene conto della probabilità di concordanza casuale. Viene calcolato in base al rapporto tra l'accordo (Veri Positivi + Veri Negativi) in eccesso rispetto alla probabilità di concordanza casuale, e l'eccesso massimo ottenibile. Valori elevati indicano risultati migliori
- *P*-Value (Acc > NIR): valore che testa l'ipotesi secondo cui l'accuratezza del modello non è migliore della soglia detta "*No Information Rate*" (NIR; Kuhn, 2008), che rappresenta la probabilità di predire correttamente un'etichetta se si sceglies-

se sempre la classe maggioritaria. Un p-value basso (< 0.05) indica che c'è una forte evidenza contro l'ipotesi nulla, suggerendo che l'accuratezza del modello è significativamente migliore del NIR

Lo scopo di questi tre indicatori è evidenziare rispettivamente: la proporzione di testi classificati correttamente, il livello di accordo tra gli autori previsti e quelli effettivi (in altre parole, la *performance* di ciascun classificatore rispetto al livello di base di un classificatore casuale) e la significatività complessiva di ciascun modello. Poiché nel *test set* le due classi hanno sempre una numerosità uguale, il NIR vale sempre 0.5, come in qualsiasi *dataset* bilanciato.

Tabella 3.1: Confronto delle *performance* di classificazione per *feature extraction*, LLM e CA con criteri di valutazione quali: *Accuracy*, Kappa di Cohen e un *test* unilaterale *P*-Value (Acc > NIR) che verifica se l'accuratezza è maggiore del NIR. La colonna *Misclassifications* (Mis.) indica il numero di testi classificati erroneamente nel *test set*. In **grassetto** le combinazioni che hanno raggiunto l'*Accuracy* massima per ogni tipologia.

Tipologia	**Modello**	**Train Control**	**Acc.**	**Mis.**	**Kappa**	***P*-Value**
Text Features	**svmRadial**	**LGOCV**	**1.000**	**0**	**1.000**	7.523e-37
Text Features	glmnet	adaptive_cv	1.000	0	1.000	7.523e-37
Text Features	ranger	adaptive_boot	1.000	0	1.000	7.523e-37
Text Features + MFW	**xgbTree**	**repeatedcv**	**1.000**	**0**	**1.000**	7.523e-37
Text Features + MFW	gbm	optimism_boot	1.000	0	1.000	7.523e-37
Text Features + MFW	gbm	adaptive_boot	1.000	0	1.000	7.523e-37
CA-15	**svmRadial**	**repeatedcv**	**1.000**	**0**	**1.000**	7.523e-37
CA-15	ranger	optimism_boot	1.000	0	1.000	7.523e-37
CA-15	gbm	adaptive_boot	1.000	0	1.000	7.523e-37
Text Features + MFW + ngrams	**glmnet**	**repeatedcv**	**0.983**	**2**	**0.966**	5.462e-33
Text Features + MFW + ngrams	svmRadialWeights	optimism_boot	0.983	2	0.966	5.462e-33
Text Features + MFW + ngrams	gbm	LGOCV	0.983	2	0.966	5.462e-33
Bert-ita-15	**svmRadial**	**repeatedcv**	**0.975**	**3**	**0.950**	2.167e-31
Bert-ita-15	svmRadial	optimism_boot	0.975	3	0.950	2.167e-31
Bert-ita-15	svmRadialWeights	adaptive_cv	0.975	3	0.950	2.167e-31
Bert-ita-150	**xgbTree**	**repeatedcv**	**0.975**	**3**	**0.950**	2.167e-31
Bert-ita-150	xgbTree	optimism_boot	0.975	3	0.950	2.167e-31
Bert-ita-150	xgbTree	adaptive_boot	0.975	3	0.950	2.167e-31

Bert-ita-768	**gbm**	**repeatedcv**	**0.967**	**4**	**0.933**	6.396e-30
Bert-ita-768	cforest	optimism_boot	0.967	4	0.933	6.396e-30
Bert-ita-768	svmRadial	repeatedcv	0.967	4	0.933	6.396e-30
Bert-multi-15	**xgbTree**	**repeatedcv**	**0.967**	**4**	**0.933**	6.396e-30
Bert-multi-15	gbm	adaptive_boot	0.958	5	0.916	1.497e-28
Bert-multi-15	glmnet	optimism_boot	0.958	5	0.916	1.497e-28
Bert-multi-768	**glmnet**	**optimism_boot**	**0.967**	**4**	**0.933**	6.396e-30
Bert-multi-768	glmnet	adaptive_boot	0.967	4	0.933	6.396e-30
Bert-multi-768	svmRadial	repeatedcv	0.958	5	0.916	1.497e-28
CA-479	**ranger**	**repeatedcv**	**0.967**	**4**	**0.933**	6.396e-30
CA-479	ranger	adaptive_boot	0.958	5	0.916	1.497e-28
CA-479	gbm	adaptive_boot	0.958	5	0.916	1.497e-28
CA-150	**ranger**	**optimism_boot**	**0.967**	**4**	**0.933**	6.396e-30
CA-150	gbm	optimism_boot	0.967	4	0.933	6.396e-30
CA-150	ranger	repeatedcv	0.958	5	0.916	1.497e-28
MFW	**rf**	**repeatedcv**	**0.958**	**5**	**0.917**	1.497e-28
MFW	ranger	adaptive_cv	0.950	6	0.900	2.897e-27
MFW	svmRadial	LGOCV	0.950	6	0.900	2.897e-27
Bert-multi-150	**gbm**	**adaptive_cv**	**0.950**	**6**	**0.900**	2.897e-27
Bert-multi-150	glmnet	repeatedcv	0.950	6	0.900	2.897e-27
Bert-multi-150	glmnet	optimism_boot	0.950	6	0.900	2.897e-27
char-grams	**glmnet**	**optimism_boot**	**0.883**	**14**	**0.767**	5.784e-19
char-grams	glmnet	adaptive_boot	0.883	14	0.767	5.784e-19
char-grams	glmnet	adaptive_cv	0.883	14	0.767	5.7845e-19
n-grams	**ranger**	**repeatedcv**	**0.850**	**18**	**0.700**	9.879e-16
n-grams	glmnet	adaptive_boot	0.850	18	0.700	9.879e-16
n-grams	svmRadial	optimism_boot	0.850	18	0.700	9.879e-16

Tramite la Tabella 3.1 è possibile osservare una panoramica delle combinazioni che hanno raggiunto i valori di *Accuracy* più elevati, ordinate per tipologia. Si può facilmente notare come alcune tipologie abbiano ottenuto un livello di accuratezza pari a 1, indicando che tutti i 120 testi contenuti nel test *set* sono stati assegnati correttamente al loro autore. È stato possibile raggiungere questo risultato nei casi di Text Features, Text Features + MFW e CA-15. La grande capacità predittiva delle Text Features indica che i modelli basati sulle caratteristiche testuali e lessicali sono estremamente efficaci nella discriminazione dei testi. Al contrario, le MFW prese singolarmente hanno ottenuto risultati peggiori, sebbene il livello di accuratezza prodotto sia comunque elevato.

Ciò non accade nel caso delle classificazioni tramite n-grammi e char-grammi che

hanno portato i valori di *Accuracy* più bassi. Queste *feature* potrebbero essere meno informative o troppo sparse per svolgere un buon compito predittivo. L'aggiunta delle MFW alle Text Features non ha variato l'accuratezza, che poteva solo diminuire, suggerendo che le caratteristiche lessicali sono sufficienti per la classificazione. Tuttavia, combinando anche gli n-grammi a queste quantità, i risultati peggiorano leggermente. È possibile che questa tipologia di dati aggiunga rumore o ridondanza, considerati anche gli scarsi valori di accuratezza nel loro utilizzo in solitario. Queste considerazioni sono in linea con quanto osservato nelle analisi esplorative tramite PCA e MDS nel Capitolo 2.

Tra le varie dimensionalità adottate per l'Analisi delle Corrispondenze, si raggiungono i risultati migliori basandosi solo sui i primi 15 assi, che forniscono una classificazione perfetta. Al contrario, usare tutti gli assi peggiora il livello di accuratezza, anche se comunque viene classificata correttamente la quasi totalità dei testi. La motivazione del perché ciò accade potrebbe risiedere nel fatto che includere troppi assi introduce del rumore nel modello, che può peggiorare la sua capacità di generalizzare su dati non visti. Un numero elevato di assi può condurre all'*overfitting*, portando il modello a adattarsi a caratteristiche specifiche del *set* di addestramento che non sono presenti nei dati di *test*. Invece, nel caso in cui si sceglie di servirsi dei primi 150 assi, i risultati non variano particolarmente. Questo probabilmente è dovuto al fatto che questo numero di componenti spiega comunque una buona parte della varianza totale, come discusso precedentemente.

Tra i tre metodi di rappresentazione dei testi, i modelli linguistici producono i risultati peggiori. Nel caso di *bert-base-italian-uncased* si classificano correttamente il 97.5% dei testi quando vengono utilizzate le versioni ridotte dei *dataset* di addestramento, mentre, servendosi delle 768 dimensioni originali, l'accuratezza scende al 96.7%. Sebbene ridurre la dimensionalità dei dati fornisca un leggero miglioramento, i valori sono comunque lontani da quelli raggiunti dagli altri metodi di vettorizzazione. Per questo motivo è necessario trovare una strada alternativa per migliorare i risultati. Inoltre, il modello multilingua classifica i testi leggermente peggio e, in questo caso, variare il numero di componenti nei dati ridotti induce una leggera differenza, ma solo peggiorando il livello di accuratezza. Si può affermare che, come da aspettative, il modello linguistico addestrato esclusivamente per la lingua italiana sia in grado di prevedere in modo migliore l'autore dei testi. Tale risultato è comprensibile, dal momento che *bert-base-multilingual-uncased* è necessariamente più generico e meno preciso.

La riduzione dei dati ha in alcuni casi aiutato a raggiungere risultati migliori, presumibilmente perché ha permesso ai modelli di concentrarsi sulle informazioni maggiormente rilevanti. Un ulteriore vantaggio consiste nel minor costo computazionale e la conse-

guente maggiore velocità di esecuzione dei modelli. Infatti, l'uso di tutte le informazioni disponibili aumenta significativamente la complessità degli algoritmi. Questo aumento non solo richiede più risorse per l'addestramento e la valutazione, ma può anche complicare l'ottimizzazione del processo. Modelli più elaborati sono più difficili da ottimizzare correttamente e possono soffrire di problemi numerici. Inoltre, troppi dati potrebbero introdurre collinearità, ovvero alta correlazione tra dimensioni o assi. Questo effetto può rendere i coefficienti meno stabili e meno precisi. Difatti, quantità ridondanti non aggiungono nuove informazioni utili ma aumentano solo la complessità in gioco. Perciò, per bilanciare i tempi di calcolo e l'accuratezza dei risultati conviene adoperare *set* di *feature* più leggere. Queste valutazioni possono essere valorizzate da una precedente ricerca esplorativa sulla quantità di varianza spiegata dai dati a diverse dimensionalità decrescenti.

Per quanto riguarda i differenti modelli utilizzati per la classificazione, non si può affermare che ci siano specificazioni che funzionano meglio di altre. Infatti, oltre a `glmnet`, sia i modelli ad albero e le loro varianti, sia le diverse tipologie di SVM hanno raggiunto l'accuratezza massima. In questo caso sarebbe opportuno scegliere di adoperare i modelli con i tempi di esecuzione inferiori come `svmRadial`. Anche la scelta del parametro di *train control* si è rivelata poco incisiva nella previsione. Questo aspetto può essere meno critico rispetto alla tipologia utilizzata. Dunque, pur agendo in modo diverso, i vari tipi di modelli sono stati in grado di cogliere gli aspetti numerici dei testi; allo stesso modo, il metodo di convalida non ha influito sulla generalizzazione nel *test set*.

In sintesi, la scelta delle tipologie di *feature* risulta essere il fattore determinante per le *performance* di classificazione, mentre la scelta del modello e del *train control* ha un impatto trascurabile. Questa valutazione suggerisce che l'attenzione principale dovrebbe essere posta sulla selezione e sulla creazione delle rappresentazioni numeriche dei testi, piuttosto che sulla complessità del modello o sulla strategia di validazione da utilizzare.

3.2.1 Il processo di Fine-Tuning

Dato che, rispetto agli altri metodi proposti, l'utilizzo dei LLM ha prodotto i risultati peggiori nel compito di classificazione dei testi, è necessario cercare di migliorare questa tecnica per ottenere prestazioni superiori. Una delle migliori strategie per raggiungere questo obiettivo è attraverso il *fine-tuning*. Con questo termine si intende una tecnica di *training* che consente a un modello pre-allenato di essere addestrato per compiti specifici o domini particolari attraverso una fase di addestramento supplementare. Mentre i LLM sono allenati su enormi quantità di dati generici per acquisire una comprensione

linguistica generale, il *fine-tuning* consente di specializzare ulteriormente il modello su un *set* di dati specifico, migliorando così la sua capacità di catturare le singolarità e le particolarità del dominio di interesse.

Durante il *fine-tuning*, si utilizzano i pesi pre-calcolati del modello come punto di partenza e si esegue l'addestramento su un nuovo *dataset*, solitamente più piccolo ma mirato, che riflette il compito specifico. I pesi iniziali vengono modificati leggermente sulla base delle quantità (i testi) in *input* e della variabile obiettivo (l'autore). Così facendo vengono generate molteplici rappresentazioni leggermente differenti tra loro, che potrebbero riflettere meglio la natura dei dati, apprendere le loro peculiarità e adattare le rappresentazioni per migliorare le prestazioni finali.

In `R` è possibile adoperare il pacchetto `grafzahl` (Chan, 2023) che si basa sulle librerie `transformers` (Wolf *et al.*, 2019) e `simpletransformers` (Rajapakse, 2022) disponibili in `Python`, in maniera simile a quanto avviene con la libreria `text`. Servendosi delle sue funzioni è possibile eseguire il *fine-tuning* di modelli linguistici pre-addestrati per migliorare le rappresentazioni testuali in base al compito di classificazione desiderato. Dal momento che dalle analisi precedenti risulta che il modello *bert-base-italian-uncased* ha ottenuto le *performance* migliori, si è deciso di ottimizzarlo eseguendo un addestramento supplementare sui testi contenuti nel *training set*. Terminata questa fase, è stata eseguita una previsione dell'autore sui dati appartenenti al *test set*. All'interno del processo vengono realizzate convalide incrociate ripetute, a seconda del numero di epoche di addestramento selezionato. Individuare il numero ottimale è fondamentale per evitare i casi di *overfitting* e *underfitting* e ottenere il bilanciamento desiderato. In aggiunta, metodi di regolarizzazione come *dropout* e *weight decay*, combinati con la tecnica dell'*early stopping*, permettono una generalizzazione dei risultati il più efficace possibile (Girosi *et al.*, 1995).

Tabella 3.2: Confronto delle matrici di confusione applicate ai risultati prodotti dalla tipologia Bert-ita-15 tramite il modello `svmRadial` con `repeatedcv` come *train control* (a sinistra) e dal modello risultante dopo il *fine-tuning* (a destra).

		Riferimento	
		GPT	HUM
Prev.	GPT	60	3
	HUM	0	57

		Riferimento	
		GPT	HUM
Prev.	GPT	60	2
	HUM	0	58

Nonostante queste accortezze, non è stato possibile raggiungere un livello di *Accuracy* pari a 1. Tuttavia, nel caso migliore sono stati classificati correttamente il 98.3% dei testi

nel *test set*. È possibile osservare i risultati più nel dettaglio analizzando le matrici di confusione del modello addestrato con il *fine-tuning* e del modello con l'accuratezza più elevata ottenuto attraverso le classificazioni precedenti.

Le matrici di confusione sono utili strumenti di valutazione adatti a misurare le prestazioni in ambito di classificazione. La Tabella 3.2 mostra il confronto tra i valori predetti dal modello e i valori reali delle classi nei due casi considerati. Il processo di *fine-tuning* ha permesso di classificare correttamente un testo in più del modello non perfezionato. In entrambi i casi tutti i testi appartenenti alla classe GPT vengono associati al loro vero autore, mentre alcuni dei testi prodotti dagli studenti risultano erroneamente attribuiti alla categoria opposta. Di conseguenza, l'uso dei modelli linguistici, pur dimostrando la capacità di produrre classificazioni di alta qualità risulta essere la tecnica meno efficace tra quelle considerate. È possibile che altre tecniche di *fine-tuning* possano migliorare i risultati, ma non sono state prese in esame in questo lavoro. Inoltre, può accadere che il numero di testi su cui eseguire questo perfezionamento sia troppo limitato per migliorare drasticamente i risultati.

3.3 Interpretazione dei risultati

I metodi e i modelli discussi fino a questo punto hanno permesso di raggiungere risultati notevoli e, in alcuni casi, livelli di *Accuracy* pari a 1, nel risolvere il problema di classificazione proposto. La fase successiva consiste nell'interpretazione dei risultati, al fine di stabilire quali *feature* e caratteristiche dei testi hanno contribuito maggiormente all'individuazione della classe di appartenenza, sia nel caso in cui fosse corretta, sia nel caso contrario.

Tuttavia, alcune delle tecniche e dei modelli utilizzati sono considerati "*black box*", a causa della loro elevata complessità e limitata trasparenza. Questo termine si riferisce a quei sistemi, a scatola nera, per cui non è noto a priori cosa contengano o come si comportino, ma è possibile studiarne il comportamento esclusivamente analizzando le risposte che essi producono (*output*), a fronte delle sollecitazioni che ricevono (*input*). Nel campo del *machine learning*, i modelli *black box* sono estremamente potenti e possono fornire previsioni accurate. Eppure, la loro opacità può essere problematica in vari contesti, specialmente quando le decisioni basate su di essi hanno implicazioni significative, o semplicemente quando si vuole capire il loro funzionamento per incrementare ulteriormente le *performance*.

L'*Explainable Machine Learning* è un campo emergente nell'ambito dell'intelligenza artificiale che si concentra sulla comprensione e interpretazione dei modelli di appren-

dimento automatico, cercando di fornire una motivazione di come e perché sono state fatte certe previsioni. I vantaggi prodotti sono di facile intuizione: una cognizione più approfondita dei processi incrementa la fiducia nelle decisioni prese dai modelli e facilita l'individuazione e la correzione degli eventuali errori presenti. Alcuni algoritmi come gli alberi decisionali sono, per loro natura, più interpretabili rispetto ai modelli complessi come le reti neurali o alle SVM. Tuttavia, metodi come LIME (*Local Interpretable Model-agnostic Explanations*; Ribeiro *et al.*, 2016) e SHAP (*SHapley Additive exPlanations*; Lundberg e Lee, 2017) aiutano a fare chiarezza e a comprendere meglio le motivazioni dietro alle scelte dei modelli. Inoltre, visualizzazione grafiche come la proiezione dei testi in uno spazio a ridotte dimensioni o grafici a barre relative all'importanza delle singole caratteristiche, possono contribuire a una più chiara interpretazione dei risultati.

3.3.1 Feature Extraction

In vari ambiti, i modelli semplici sono spesso preferiti per la loro facilità di interpretazione, anche se possono essere meno accurati di quelli complessi. Nel caso di questo lavoro, il metodo basato sull'estrazione delle caratteristiche si è rivelato non solo estremamente efficace, ma probabilmente è anche il più facile da ispezionare. Come preannunciato, la possibilità di interpretare correttamente l'*output* di previsione di un modello è quasi più importante del livello di accuratezza stesso. Nonostante gli algoritmi ad albero e gli altri modelli proposti non siano quasi mai comprensibili nella loro interezza, la crescente disponibilità di metodi per affrontare questo problema sta portando in primo piano il compromesso tra *performance* e interpretabilità dei risultati. Numerosi approcci sono stati proposti a questo scopo e la maggior parte di essi si riconduce a ricostruire il contributo delle singole variabili del modello nella creazione della previsione.

L'importanza delle variabili

La funzione `varImp` della libreria `caret` in `R` permette di ordinare le variabili esplicative di un modello secondo la misura della loro importanza per la previsione finale. Dato che non tutte le tipologie di modello dispongono di una scomposizione delle previsioni a livello dei singoli predittori, si è deciso di esaminare solo i casi che consentono il calcolo di questa metrica. Per i modelli `ranger` (implementazione di *Random Forest*) la misura dell'importanza di ciascuna variabile in ogni albero è ottenuta tramite la procedura descritta nell'Algoritmo 1, per il quale, se la permutazione di una variabile riduce significativamente l'accuratezza predittiva, la variabile è considerata importante. I modelli che comprendono il *boosting*, come `xgbTree` e `gbm`, utilizzano un approccio simile a quello

dei singoli alberi, ma con una differenza significativa: l'importanza delle variabili viene sommata su tutte le iterazioni del processo. In questo modo si valorizzano le variabili più importanti nel complesso.

Algoritmo 1 Calcolo dell'importanza delle variabili in un modello *Random Forest*

Input: Dati di addestramento X, variabili V
Addestra il modello *Random Forest* su X.
for ogni albero nella foresta **do**
    Seleziona i dati *out-of-bag* per l'albero;
    Calcola l'accuratezza predittiva sui dati *out-of-bag*.
    for ogni variabile v in V **do**
        Permuta i valori di v nei dati *out-of-bag*;
        Ricalcola l'accuratezza predittiva sui dati permutati;
        Registra la differenza tra l'accuratezza originale e quella dopo la permutazione.
    end for
end for
Media le differenze di accuratezza su tutti gli alberi per ciascuna variabile;
Normalizza le differenze per la deviazione standard.
Output: Importanza delle variabili V

Nel seguito, si è scelto di esaminare l'importanza delle variabili per le combinazioni che in base alla Tabella 3.1 hanno raggiunto i risultati migliori. Vengono presi in considerazione casi appartenenti a tipologie differenti in modo da avere un quadro più ampio delle *feature* utilizzabili.

La Figura 3.3 mostra che le Text Features a cui viene attribuita un'importanza maggiore sono *perplexity*, `sd_token_per_frase` e *burstiness*. La loro importanza non cambia anche quando tra i predittori vengono inserite le MFW e gli n-grammi, stando a indicare la loro robustezza in seguito alla variazione delle specificazioni dei modelli. Come descritto nel Capitolo 2, la *perplexity* indica la complessità, la variabilità e l'instabilità dei testi e, similmente, la *burstiness* misura la dispersione relativa delle frequenze dei token nei testi, mentre `sd_token_per_frase` mostra la variabilità della lunghezza delle frasi all'interno di essi. Queste tre metriche sono accomunate dal fatto che ognuna di esse riflette un particolare aspetto dell'imprevedibilità del testo su cui è calcolata.

L'elevata capacità predittiva conferma il loro essere in grado di fornire informazioni complementari e non ridondanti rispetto ad altri predittori. La *perplexity*, ad esempio, permette di comprendere meglio la complessità linguistica di un testo, la *burstiness* aiuta a identificare schemi di utilizzo non uniformi che sono spesso indicativi di una scrittura più spontanea o meno strutturata. Infine, `sd_token_per_frase`, fornisce ulteriori dettagli sulla struttura sintattica e sulla fluidità del discorso. La loro capacità di rivelare

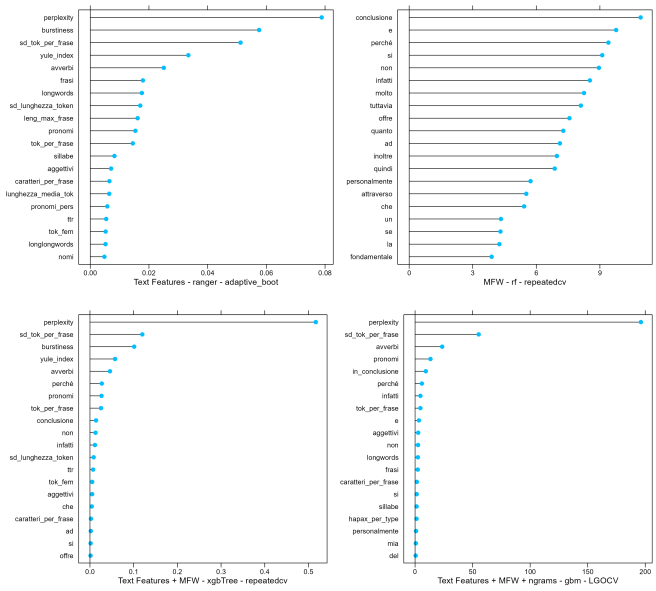


Figura 3.3: Prime 20 variabili per importanza predittiva in alcuni dei modelli con *Accuracy* più elevata, appartenenti a tipologie diverse. L'importanza tra modelli diversi è calcolata su scale differenti.

aspetti distintivi del testo rende queste *feature* particolarmente efficaci per l'analisi del riconoscimento d'autore.

In aggiunta, analizzare le MFW più importanti può aiutare a comprendere le tendenze lessicali e stilistiche più generali. Avendo selezionato solo parole vuote come predittori esse compongono l'intera graduatoria e, in particolare, "conclusione", "e", "perché", "infatti" si rilevano i token più significativi per la previsione, mentre l'unico n-gramma in grado di rientrare tra le variabili complessivamente più importanti è il bi-gramma "in_conclusione". L'analisi delle MFW rivela che queste parole funzionali sono fondamentali per identificare le caratteristiche stilistiche dei testi. Parole come "conclusione", "perché" e "infatti" possono essere indicative di strutture argomentative e che variano tra i due autori. Inoltre, l'uso frequente di "conclusione" può suggerire una preferenza per una chiusura esplicita dei discorsi. Quando le Text Features sono incluse nel modello, l'importanza delle MFW diminuisce, suggerendo che le prime catturano informazioni più complesse e dettagliate rispetto alle semplici frequenze relative delle parole vuote. Ciò è evidenziato dalla sostituzione del bi-gramma "in_conclusione" alla MFW "conclusione", indicando che le combinazioni di parole possono fornire contesti più ricchi e significativi rispetto ai singoli token.

Questo approccio integrato permette di sfruttare al meglio le informazioni disponibili, migliorando la capacità dei modelli di prevedere correttamente l'origine dei testi e offrendo preziosi spunti per la comprensione delle dinamiche di scrittura. Tuttavia, queste informazioni risultano essere parziali e incomplete se non si conosce la direzione dell'influenza delle *feature* nelle previsioni. Sapere quale variabile è importante costituisce solo una parte dell'analisi: è cruciale anche comprendere come ciascun predittore influenza i risultati, ovvero capire se un aumento (o una diminuzione) dei valori di ciascuna variabile porta a una previsione più probabile verso quale autore.

Il metodo SHAP

Il metodo SHAP permette di spiegare le singole previsioni effettuate dai modelli di *machine learning* basandosi sugli SHAP Values, provenienti dalla teoria dei giochi di collaborazione (Molnar, 2023). Questi valori rappresentano dei concetti di soluzione utilizzati per assegnare una ricompensa ad ogni giocatore presente in una coalizione, in funzione del contributo marginale che apporta ad essa. La teoria si basa sull'ipotesi che una previsione possa essere spiegata assumendo che il valore di una certa *feature* (variabile esplicativa), per una determinata unità statistica, sia un "giocatore" in un gioco dove la previsione è la ricompensa.

Con queste premesse, il Valore di Shapley è un sistema per distribuire la ricompensa ottenuta dalla coalizione tra i suoi componenti. Lo scopo che si prefigge è di suddividere tale ricompensa in modo proporzionale al contributo che ogni giocatore apporta alla coalizione. Secondo questa interpretazione, il gioco è il problema di classificazione per un singolo testo del *dataset* e i giocatori sono i valori delle *feature* che collaborano per ricevere una ricompensa, ovvero la previsione della risposta. L'obiettivo è spiegare la differenza, chiamata guadagno, tra il valore della singola previsione e la previsione media per tutti i testi considerati.

In un modello lineare

$$f(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

dove x è l'unità statistica per la quale si vuole calcolare la previsione, ogni x_j è il valore di una *feature* j, per $j = 1, \ldots, p$ e β_j è il peso relativo alla *feature* j; il contributo ϕ_j della j-esima *feature* nella previsione $\hat{f}(x)$, ovvero il contributo che ogni giocatore apporta alla coalizione, è pari a:

$$\phi_j(\hat{f}) = \beta_j x_j - E(\beta_j X_j) = \beta_j x_j - \beta_j E(X_j)$$

con $E(\beta_j X_j)$ effetto medio stimato per la j-esima *feature*. Questa quantità corrisponde dunque alla differenza tra l'effetto singolo e l'effetto medio della *feature*. Inoltre, se si sommano tutti i contribuiti dei predittori per una unità statistica x, si ottiene:

$$\begin{aligned}\sum_{j=1}^{p} \phi_j(\hat{f}) =& \sum_{j=1}^{p} (\beta_j x_j - E(\beta_j X_j)) \\ =& (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} E(\beta_j X_j)) \\ =& \hat{f}\,(x) - E(\hat{f}\,(X))\end{aligned}$$

che corrisponde al valore previsto per l'osservazione considerata meno il valore medio di previsione, cioè il guadagno.

Per poter generalizzare questa interpretazione ad ogni tipo di modello, accorre in aiuto la teoria dei giochi: il Valore di Shapley è definito attraverso una funzione valore *val* dei giocatori nella coalizione S (Aas *et al.*, 2021). Il Valore di Shapley $\phi_j(val)$ associato al singolo valore di una *feature* è il suo contribuito alla ricompensa, pesato e sommato tra tutte le possibili combinazioni dei valori della *feature*:

$$\phi_j(val) = \sum_{S \subseteq \{1,\dots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} (val\,(S \cup \{j\}) - val(S))$$

dove S è un sottoinsieme delle p *feature* usate nel modello. $val_x(S)$ è la previsione per i valori delle *feature* nella coalizione S, integrati rispetto alle *feature* non incluse in S:

$$val_x(S) = \int \hat{f}\,(x_1, \dots, x_p)\, d\mathbb{P}_{x \notin S} - E_X(\hat{f}\,(X))$$

Si ottengono diverse coalizioni permutando casualmente i valori di una *feature* a turno tra quelle considerate e ripetendo il processo numerose volte. Per questo motivo, anche al di fuori del modello lineare, il Valore di Shapley associato al valore di una determinata *feature* j indica che quella *feature* ha fornito un contributo alla previsione pari a ϕ_j per una determinata unità statistica, rispetto alla previsione media per l'intero *dataset*.

Come anticipato, capire come e quanto le singole variabili influenzano la previsione finale è semplice per un modello di regressione lineare, ma diventa più complesso negli altri casi. Tuttavia, una innovazione introdotta dagli SHAP Values è che il loro valore può essere rappresentato come un metodo additivo di attribuzione delle caratteristiche, in modo simile a quanto avviene in un modello lineare, facilitando l'interpretazione. Infatti,

SHAP specifica i contribuiti alle previsioni come:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

dove g è il modello esplicativo, $z' \in \{0,1\}^M$ è il vettore di coalizione che indica la presenza o l'assenza dei valori ϕ_j, M è la grandezza massima della coalizione e $\phi_j \in \mathbb{R}$ è il valore attribuito alla *feature* j, ovvero lo SHAP Value.

Al fine di arrivare a questa rappresentazione con i dati a disposizione, è necessario innanzitutto scomporre l'importanza delle Text Features all'interno del primo modello considerato, anche se a causa di ragioni computazionali vengono analizzate le previsioni effettuate tramite una sua versione ridotta. Per fare ciò si crea un modello `ranger` basato solo sui cinque predittori più importanti secondo `varImp`, per attribuire l'autore ai testi nel *test set*. Successivamente, tramite la libreria `shapr` (Sellereite *et al.*, 2024) si calcolano gli SHAP Values per interpretare l'*output* di previsione. Il calcolo viene effettuato attraverso un approccio empirico, dal momento che non si può assumere normalità per la distribuzione dei valori delle *feature*.

La funzione richiede di specificare un valore di previsione medio per i dati presenti nel *test set*, che viene posto pari a 0.5 siccome il *dataset* è bilanciato. Questa quantità indica il valore medio atteso per la variabile risposta in assenza di predittori. A livello interpretativo, una previsione con probabilità maggiore o uguale a 0.5 viene attribuita a GPT, mentre negli altri casi all'etichetta HUM, come nelle analisi precedenti. In aggiunta, viene creato un secondo modello ridotto basato esclusivamente sulle cinque MFW più rilevanti secondo `varImp`. Le previsioni e il calcolo degli SHAP Values sono effettuati in modo analogo a quanto fatto per le Text Features.

Inizialmente si analizza l'influenza complessiva delle variabili nella determinazione delle previsioni per il *test set*. Questa importanza globale I_j, specifica per ogni *feature* j, si ottiene calcolando la media dei valori assoluti degli SHAP Values $\phi_j^{(i)}$ associati a j per tutti gli n testi, ed è proporzionale alla quantità ottenuta: maggiore è lo SHAP Value assoluto, maggiore è l'influenza della variabile.

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j^{(i)}|$$

Per quanto riguarda le Text Features in Figura 3.4, l'ordinamento risulta simile a quello prodotto da `varImp`: la *perplexity* rimane la variabile più importante, mentre `sd_tok_per_frase` e *burstiness* si collocano subito dopo. L'importanza delle caratteri-

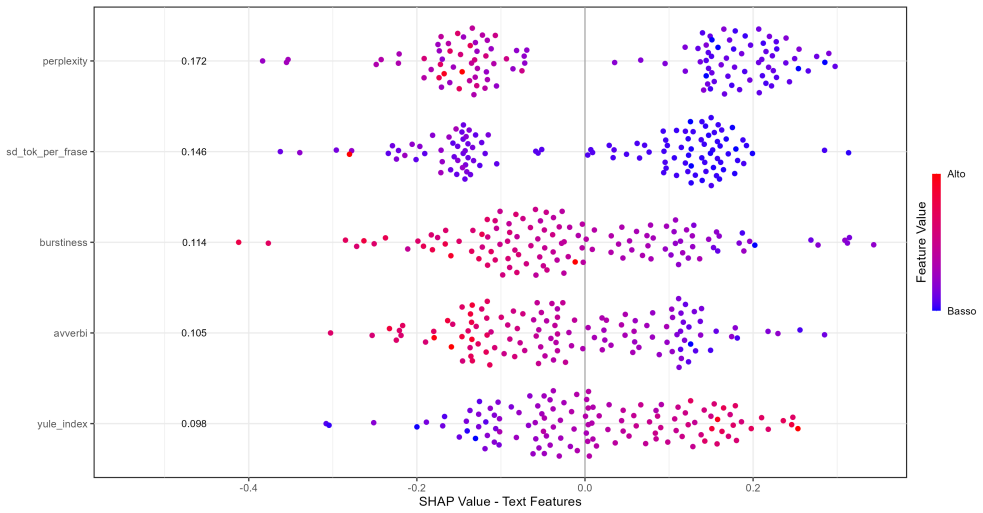


Figura 3.4: Distribuzione degli SHAP Values relativi alle principali Text Features per importanza, analizzate per ogni testo nel *test set*. L'importanza globale per ogni *feature* è indicata dai numeri alla destra del nome. Ogni punto rappresenta un valore di SHAP per una variabile esplicativa relativa alla previsione per un testo: la posizione sull'asse verticale è determinata dalla *feature* e quella sull'asse orizzontale dallo SHAP Value. Il colore dei punti indica il valore delle *feature* in una scala da basso ad alto.

stiche basata sulla permutazione è simile in termini di risultati ma possiede una grande differenza: si fonda sulla diminuzione delle prestazioni del modello. Al contrario, gli SHAP Values si basano sulla grandezza delle attribuzioni (i contribuiti) assegnate alle variabili. Un'analisi a livello globale è importante ma non contiene informazioni precise sulle singole previsioni.

Per il modo in cui è costruito, il grafico permette di osservare la relazione tra il valore di una variabile e l'impatto sulla previsione. Considerando che valori positivi degli SHAP Values tendono a spostare la previsione verso ChatGPT e valori negativi verso Human, la disposizione dei punti non fa altro che confermare le analisi del capitolo precedente. Nel caso della *perplexity* vengono formati due raggruppamenti ben distinti dove quello costituito da valori positivi è associato a valori più bassi della variabile, mentre accade il contrario per quello costituito da valori negativi. Questo significa che bassi valori di *perplexity* spingono la previsione verso l'etichetta GPT, mentre valori più elevati propendono verso HUM; ciò è in linea con i risultati precedenti. Anche per `sd_tok_per_frase` i due raggruppamenti sono ben distinguibili ma i colori risultano poco diversi tra loro. Tale fenomeno è dovuto allo scarso *range* di valori della variabile che quindi può assumere tonalità meno differenti.

Ad ogni modo, anche in questo caso SHAP Values positivi sono associati a colori leggermente più scuri, indicando valori più bassi delle *feature*. Anche questo è coerente con le attese, visto che ci si aspetta meno variabilità nella composizione delle frasi da parte di ChatGPT. Nel caso delle altre tre variabili i *cluster* sono meno netti e infatti la loro importanza è minore; ma, anche per quanto riguarda la *burstiness*, valori più elevati per il predittore sono associati ai testi prodotti dagli studenti.

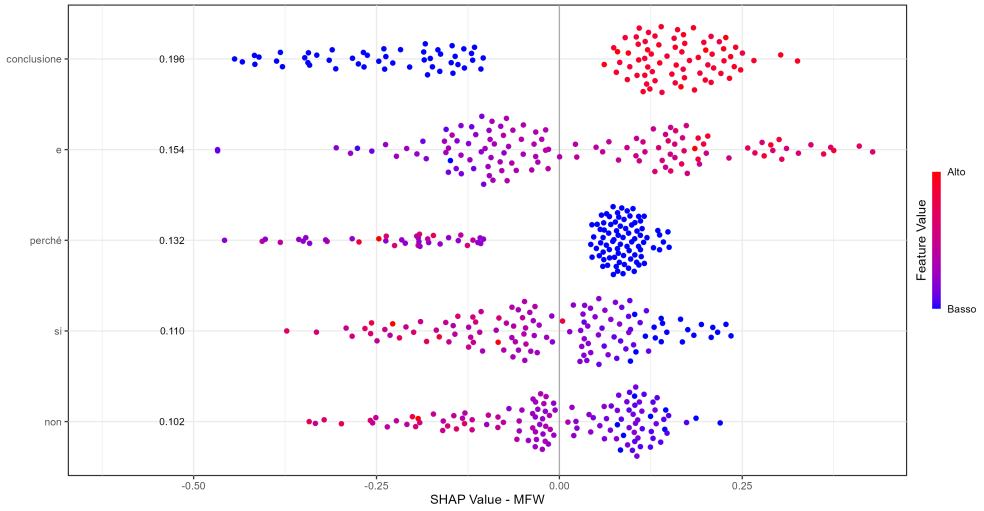


Figura 3.5: Distribuzione degli SHAP Values relativi alle principali MFW per importanza, analizzate per ogni testo nel *test set*. L'importanza globale per ogni *feature* è indicata dai numeri alla destra del nome. Ogni punto rappresenta un valore di SHAP per una variabile esplicativa relativa alla previsione per un testo: la posizione sull'asse verticale è determinata dalla *feature* e quella sull'asse orizzontale dallo SHAP Value. Il colore dei punti indica il valore delle *feature* in una scala da basso ad alto.

Per quanto riguarda le MFW in Figura 3.5, il loro ordinamento riflette perfettamente quello ottenuto tramite l'approccio `varImp`, ponendo al primo posto il token "conclusione". Gli SHAP Values relativi a questa variabile si dispongono chiaramente in due *cluster* distinti tra loro e il loro valore permette di dedurre che frequenze relative elevate della parola all'interno dei testi sono associate a testi prodotti da ChatGPT. Anche per il token "perché" la divisione è netta ma l'interpretazione è opposta visto che la sua presenza contribuisce in modo negativo alla previsione che quindi avrà valori più bassi, indicando un testo scritto da uno studente. Le restanti MFW forniscono ulteriori indicazioni meno determinanti, ma che è possibile interpretare in modo analogo alle precedenti.

Paragonate alle Text Features, le MFW sembrano discriminare meglio i due autori a livello grafico; tuttavia, i risultati dei modelli di classificazione sono inferiori quando

ci si basa esclusivamente su questa tipologia di predittori. Una possibile spiegazione del perché ciò accada può essere che i token "conclusione" e "perché" non sono presenti in tutti i testi e quindi non sempre possono fornire una indicazione precisa. Quando invece appaiono, oppure le loro frequenze sono elevate, contribuiscono in modo determinante all'interpretazione delle previsioni. Al contrario, *stopwords* come "e", "si" e "non" possono avere un peso minore nell'influenzare la scelta ma la loro elevata presenza nel corpus (il token "e" è il più presente nei testi AI e il secondo più presente in quelli HUM) le rende importanti.

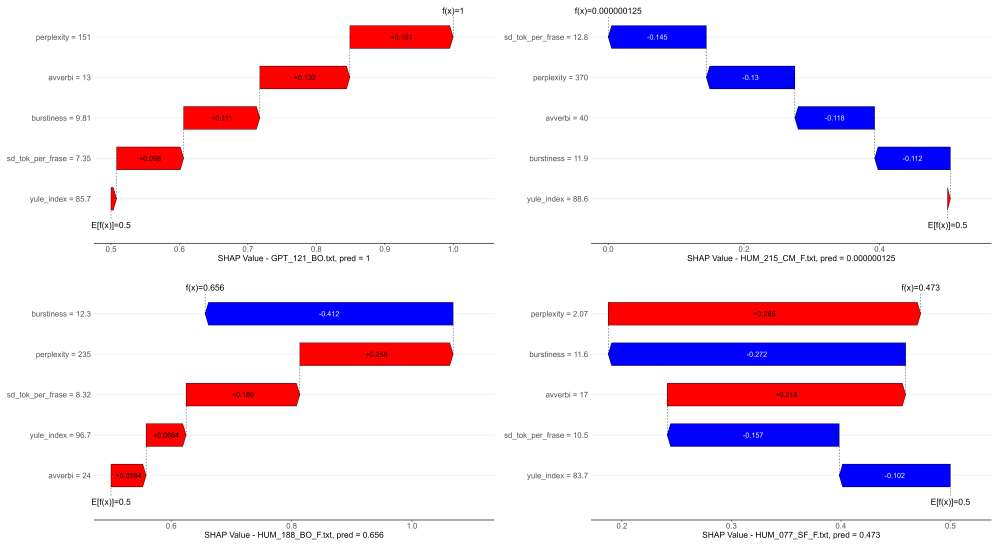


Figura 3.6: Contribuiti alle singole previsioni da parte delle Text Features nei quattro testi selezionati. In figura è indicata l'unità statistica considerata e il valore della previsione.

Il vero vantaggio interpretativo degli SHAP Values è costituito dalla possibilità di analizzare le previsioni per i singoli testi, al fine di comprendere come le *feature* hanno contribuito al raggiungimento del valore predittivo finale. Nell'analisi si sceglie di focalizzarsi su quattro specifici testi: il primo, attribuito correttamente a GPT con variabile risposta pari a 1 (testo GPT_121_BO); il secondo, attribuito correttamente a HUM con valore finale pari a 0 (testo HUM_215_CM_F); il terzo, attribuito erroneamente a GPT ma con poca sicurezza (testo HUM_188_BO_F) e infine l'ultimo, attribuito correttamente a HUM ma in modo incerto (testo HUM_077_SF_F). In questo modo sono state selezionate due previsioni corrette considerate certe dal modello, una previsione errata e una previsione corretta ma in prossimità del valore di soglia pari a 0.5.

Questo confronto permette una visione più completa del funzionamento degli SHAP

Values e consente di osservare come la contrapposizione dei valori determina la decisione finale, come mostrato in Figura 3.6. In particolare, il modello classifica correttamente 119 testi su 120, e non l'intero *test set*, per via della sua specificazione ridotta tramite cinque sole variabili esplicative. Nel caso del testo classificato con previsione finale pari a 1, tutte le *feature* vanno nella stessa direzione con un contributo decrescente dato dall'ordine nel grafico. Anche per il secondo testo l'interpretazione è simile: quasi ogni *feature* contribuisce in modo negativo al valore finale, risultando in un *output* pari a 0 che attribuisce il testo alla categoria HUM con elevata sicurezza.

Più interessante è il caso del testo classificato erroneamente: la previsione per la variabile risposta è di poco superiore alla soglia 0.5, definendo il testo come generabile dall'intelligenza artificiale. Le *feature* che spingono in questa direzione sono *perplexity* e `sd_tok_per_frase` che hanno un valore molto basso e, in aggiunta, il numero di avverbi nel testo assieme all'indice di Yule. Al contrario, i valori elevati di *burstiness* propendono correttamente per considerare il testo scritto da uno studente, ma in modo non sufficiente a correggere la previsione. L'interpretazione di `HUM_077_SF_F` è simile: viene classificato correttamente ma con poca sicurezza. In questo caso, il basso valore di *perplexity* e il numero di avverbi fanno pensare a un testo scritto da ChatGPT. Gli alti valori di *burstiness* e `sd_tok_per_frase` hanno verso opposto, contribuendo a un valore finale di poco al sotto della soglia che permette di classificare correttamente il testo.

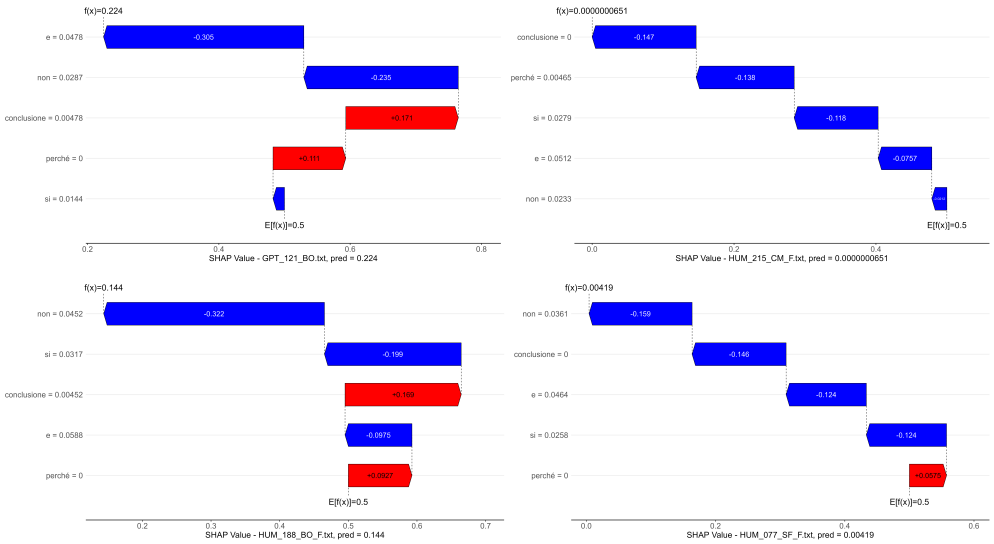


Figura 3.7: Contributi alle singole previsioni da parte delle MFW nei quattro testi selezionati. In figura è indicata l'unità statistica considerata e il valore della previsione.

Per rendere il confronto più significativo si è scelto di analizzare gli stessi singoli testi anche per le previsioni effettuate attraverso le MFW. In questo caso il modello classifica peggio i testi, confermando la minore capacità di discriminazione degli autori, come affermato in precedenza. In particolare, il testo `GPT_121_BO` viene erroneamente attribuito alla categoria HUM, mentre negli altri tre casi la previsione è corretta ed anche con elevata sicurezza. Osservando i versi dei contributi delle MFW in Figura 3.7 si può affermare che l'assenza della parola "conclusione" diminuisca notevolmente la probabilità che un testo sia stato scritto dall'intelligenza artificiale, mentre frequenze relative anche molto basse la fanno aumentare. Questo notevole divario spiega la distribuzione dei punti presente in Figura 3.5. Al contrario, in tutti i casi considerati, l'assenza del token "perché" indica che un testo è stato probabilmente prodotto da ChatGPT. Per quanto riguarda le altre tre MFW l'interpretazione è meno chiara, ma servendosi della Figura 3.5 si può dedurre che frequenze relative elevate di "sì" e "non" fanno tendere la previsione verso valori finali più vicini allo 0, mentre accade il contrario per il token "e".

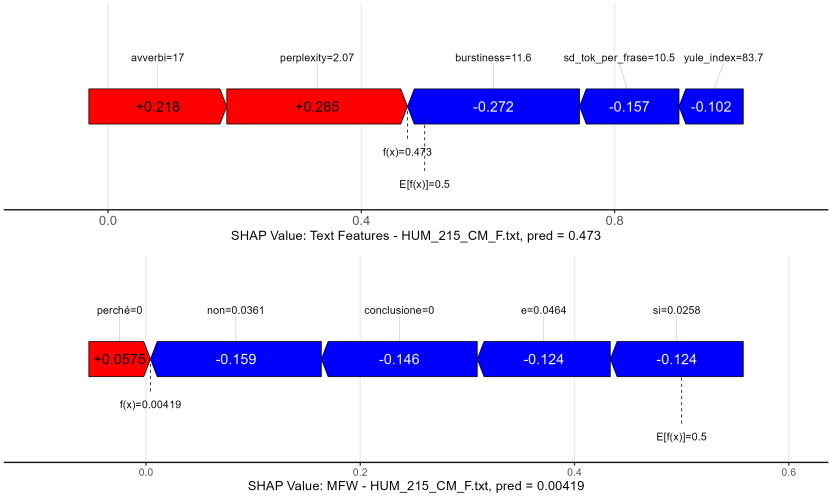


Figura 3.8: Scomposizione delle forze che contribuiscono alla composizione della previsione finale, ponendo l'enfasi sulla loro struttura oppositiva. È stato preso in considerazione un testo tra i precedenti, analizzato attraverso le Text Features e le MFW.

Come ulteriore interpretazione visiva per rafforzare il concetto di struttura additiva delle previsioni, il valore di SHAP per ogni *feature* può essere visto come una forza che può incrementare o ridurre il valore finale. Il punto di partenza è il valore di riferimento, in questo caso posto pari a 0.5. Nel grafico ogni SHAP Value è una freccia che spinge la previsione in un verso dettato dal suo segno. Queste forze si bilanciano a vicenda per produrre la previsione finale per la singola unità statistica. Focalizzandosi sul testo

HUM_215_CM_F si vede come la previsione inizialmente abbia un valore atteso pari a 0.5 e come i valori delle *feature* aumentino o decrementino questo valore in modo additivo, per giungere al risultato finale pari a 0.473. Allo stesso modo per l'analisi attraverso le MFW ogni token contribuisce alla previsione finale modificando il valore iniziale di 0.5.

Per quanto detto, gli SHAP Values si fondano su una solida base teorica relativa alla teoria dei giochi; attraverso di essi la previsione viene equamente distribuita tra i valori delle variabili esplicative, per giungere ad una rappresentazione oppositiva che confronta le singole previsioni con la previsione media. Inoltre, con questo metodo le interpretazioni globali sono consistenti con quelle locali visto che i valori di SHAP sono solo le singole unità delle spiegazioni complessive. In sintesi, SHAP rappresenta uno strumento potente e teoricamente solido per interpretare modelli di *machine learning*, facilitando una comprensione approfondita delle loro dinamiche interne e migliorando la fiducia e l'affidabilità nelle loro applicazioni pratiche.

Il metodo LIME

Tra le recenti proposte disponibili in letteratura è presente anche LIME: un metodo di interpretazione dei risultati dei modelli di *machine learning*, basato sulla loro approssimazione locale attorno a una determinata previsione. Dietro al funzionamento di questa tecnica vi è l'assunto per cui ogni modello complesso è lineare in un intorno locale. Infatti, sebbene sia difficile da dimostrare, ci si aspetta che due osservazioni molto simili si comportino in modo prevedibile anche in un modello complesso. LIME porta questo assunto alla sua naturale conclusione, affermando che è possibile adattare un modello semplice attorno a una singola osservazione che imiti localmente (e non necessariamente a livello complessivo) il comportamento del modello globale (Molnar, 2022). Questo tipo di accuratezza è chiamata fedeltà locale.

Matematicamente, il modello locale con vincolo di interpretabilità può essere espresso come segue:

$$local\,(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

Si tratta quindi del modello che minimizza la funzione di perdita $\mathcal{L}$, la quale misura quanto la spiegazione si avvicina alla previsione del modello originale f, mantenendo bassa la complessità $\Omega\,(g)$ del modello semplice. G rappresenta la famiglia dei possibili modelli (ad esempio i modelli di regressione lineare), mentre la misura di prossimità π_x indica quanto è grande l'intorno dell'osservazione x.

Il modello più semplice può quindi essere utilizzato per spiegare le previsioni del modello più complesso, ovvero il suo comportamento. Per giungere a questo obiettivo LIME adotta l'approccio descritto nell'Algoritmo 2.

Algoritmo 2 Procedura di spiegazione per una previsione tramite LIME.

Input: Modello semplice g, modello complesso f, osservazione x, numero di permutazioni N, numero di *feature* m
Output: Pesi delle m *feature* come spiegazione per il comportamento locale di $f(x)$
1. Permuta N volte x per generare $x_1, x_2, \ldots, x_N$
2. Usa f per prevedere gli esiti $f(x_1), f(x_2), \ldots, f(x_N)$
3. Calcola la distanza tra ogni x_i e l'osservazione originale x
4. Converti le distanze in punteggi di similarità
5. Seleziona un numero ristretto di m *feature* che meglio descrivono $f(x_i)$
6. Adatta g alle osservazioni permutate usando le m *feature* selezionate
7. Estrai i coefficienti delle m *feature* da g e usali come spiegazione locale per $f(x)$

In particolare, il modello semplice tenta di spiegare l'esito del modello complesso, ponendo un'enfasi maggiore sulle osservazioni permutate che sono più simili all'osservazione originale; così facendo, i pesi finali rappresentano l'importanza delle diverse *feature* nel contesto locale dell'osservazione originale. Questa procedura dipende molto dalle scelte effettuate, a partire dal modello semplice che si vuole utilizzare, ma anche in base a come vengono create le permutazioni, come sono calcolate le distanze e soprattutto quali variabili predittive vengono selezionate.

In questo caso si cerca di creare delle previsioni analoghe a quanto fatto con il metodo SHAP, in modo da poter effettuare un confronto tra le due tipologie di interpretazione. Per questo motivo, viene utilizzato lo stesso modello (complesso) predittivo creato tramite le Text Features, selezionando dai dati di *training* le variabili su cui eseguire l'addestramento. LIME dispone di diversi criteri per la selezione dei predittori su cui interpretare i dati; ma, per assicurarsi che vengano utilizzate le stesse dell'analisi precedente, si limita la scelta alle cinque adoperate lavorando con gli SHAP Values imponendo il vincolo di usarle tutte. In questo modo non avviene una vera e propria ricerca dei predittori migliori ma si sfruttano i risultati precedenti sull'importanza delle variabili.

Per quanto riguarda le altre operazioni: le permutazioni vengono campionate dalle distribuzioni delle variabili nel *training set* in modo da essere indipendenti dalla variabile spiegata, mentre per la distanza si utilizza la distanza Euclidea che viene poi convertita in misura di similarità. Infine, l'unico requisito per il modello semplice è che possa ricevere in *input* dati pesati e che sia facile da interpretare. Sebbene diversi tipi di modelli rispondano a questi criteri, LIME utilizza una regressione *Ridge* come modello di scelta.

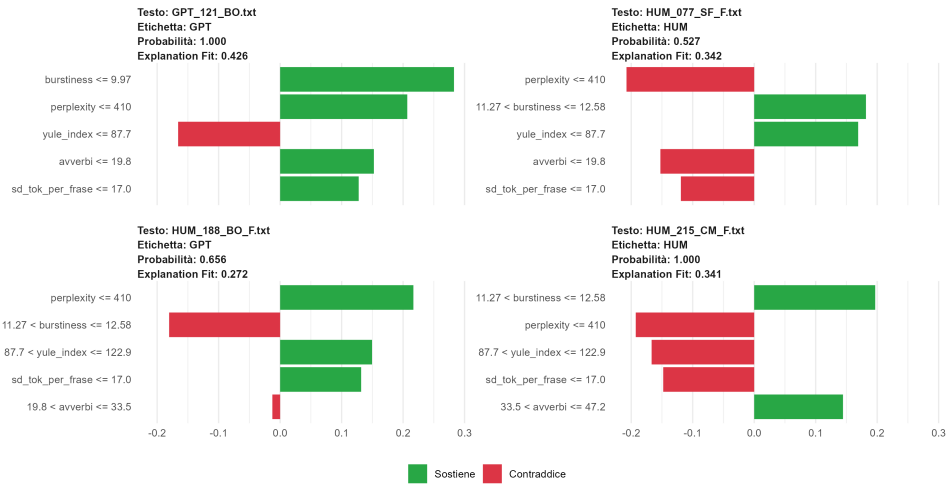


Figura 3.9: Contribuiti alle singole previsioni da parte delle Text Features nei quattro testi selezionati. In figura è indicata l'unità statistica considerata, il valore della previsione e un coefficiente di bontà di adattamento del modello.

Questo metodo può essere utilizzato per spiegare le previsioni relative al *test set* e, per coerenza, si sceglie di focalizzarsi sui quattro testi selezionati in precedenza. L'interpretazione proposta è leggermente differente da quella relativa agli SHAP Values: nella Figura 3.9 vengono mostrati i singoli contributi delle variabili alle previsioni ma l'orientamento è in questo caso riferito all'etichetta più probabile, ovvero alla classe con la probabilità più alta per l'unità statistica. Se un testo viene previsto come più probabilmente scritto da ChatGPT, i valori di alcuni dei suoi predittori sosterranno questa scelta incrementando il valore della risposta, e altri si contrapporranno propendendo per il verso opposto. Anche in questo caso, il valore assoluto del peso per ogni variabile è un indicatore della sua importanza all'interno della previsione.

Avendo utilizzato lo stesso modello adoperato con il metodo SHAP, le probabilità sono le stesse ma i contributi delle variabili possono essere differenti. Per il testo GPT_121_BO, l'indicatore `yule_index` spinge la previsione verso la categoria HUM, mentre nell'interpretazione precedente aveva un contributo positivo anche se molto piccolo. Per quanto riguarda il testo HUM_215_CM_F, *perplexity* e `sd_tok_per_frase` hanno valori bassi ritenuti più appropriati a un testo generato dall'intelligenza artificiale. In HUM_188_BO_F l'interpretazione è molto simile all'altro metodo, tranne che per la variabile avverbi: secondo gli SHAP Values ha un basso peso che aumenta il valore della previsione, mentre secondo LIME ha un basso peso che propende verso la categoria HUM. L'alto valore di *burstiness* è un forte indicatore in entrambi casi della scrittura del testo

da parte di uno studente. Infine, per HUM_077_SF_F quattro variabili su cinque sono interpretate allo stesso modo da entrambi i metodi, ad eccezione di `sd_tok_per_frase`.

Inoltre, LIME fornisce un'indicazione aggiuntiva sul valore dei predittori, indicando un intervallo oppure una soglia sopra o sotto la quale il valore viene considerato alto o basso (ad esempio nel caso del token "perché", un'alta frequenza relativa maggiore di 0.00459 viene associata a testi HUM). Nel complesso, per la maggior parte dei casi le variabili propendono verso la stessa classe della risposta sia per LIME, sia per gli SHAP Values, producendo una interpretazione fiduciosa e consistente. Tuttavia, guardando i valori di *Explanation Fit* nei quattro casi, essi non sono molto alti. Questo coefficiente simile all'R^2 fornisce un'indicazione sulla bontà di adattamento del modello e, dato che i valori sono abbastanza bassi, alcune spiegazioni degli *output* potrebbero essere poco precise.

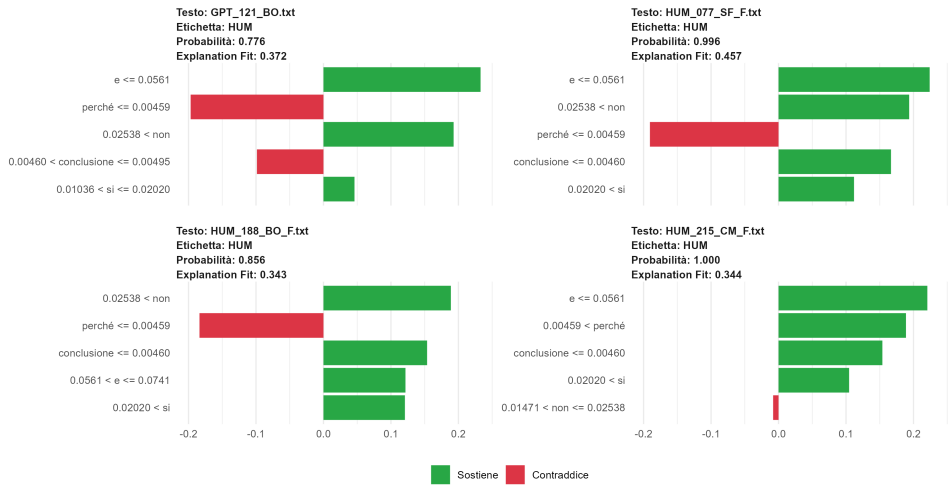
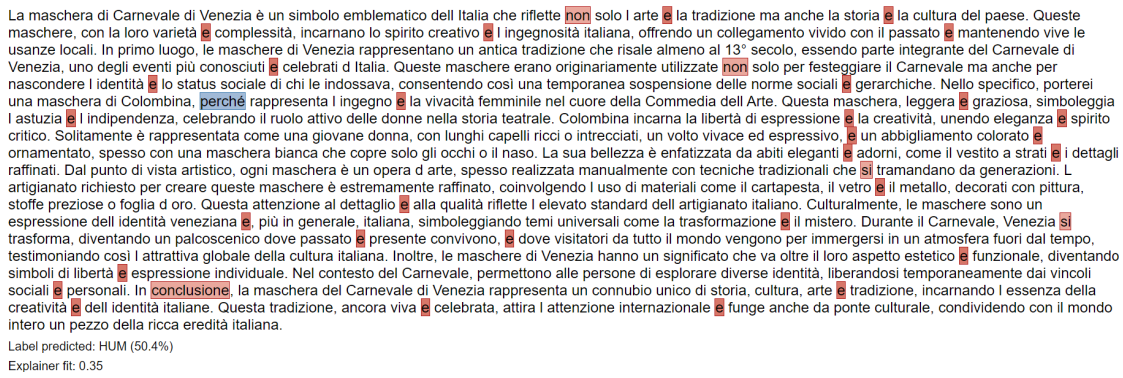


Figura 3.10: Contribuiti alle singole previsioni da parte delle MFW nei quattro testi selezionati. In figura è indicata l'unità statistica considerata, il valore della previsione e un coefficiente di bontà di adattamento del modello.

La stessa procedura viene ripetuta per le MFW partendo dalla selezione delle variabili fino ad arrivare alla scomposizione e interpretazione delle previsioni. Per il testo GPT_121_BO, la presenza del token "conclusione" conferma la sua spinta verso i testi prodotti da ChatGPT e l'assenza della parola "perché" propende nella stessa direzione, in modo analogo a quanto visto per gli SHAP Values. Anche per i testi HUM_215_CM_F e HUM_077_SF_F i due metodi forniscono spiegazioni molto simili. Nel caso del quarto testo, l'assenza del token "perché" continua a indicare che il testo potrebbe essere stato scritto da ChatGPT, mentre, al contrario di quanto ricavato tramite il metodo SHAP, la presenza della parola "conclusione" aumenta la probabilità che il testo appartenga alla

categoria HUM. Sebbene i valori di adattamento dei modelli siano leggermente superiori a quelli relativi alle Text Features, il testo HUM_188_BO_F ha l'*Explanation Fit* più basso e perciò l'interpretazione anomala potrebbe essere dovuta anche a questo.

Le considerazioni sulle MFW si possono riassumere visualizzando la loro presenza e frequenza in un testo tra quelli del corpus, come rappresentato in Figura 3.11.


La maschera di Carnevale di Venezia è un simbolo emblematico dell Italia che riflette non solo l arte e la tradizione ma anche la storia e la cultura del paese. Queste maschere, con la loro varietà e complessità, incarnano lo spirito creativo e l ingegnosità italiana, offrendo un collegamento vivido con il passato e mantenendo vive le usanze locali. In primo luogo, le maschere di Venezia rappresentano un antica tradizione che risale almeno al 13° secolo, essendo parte integrante del Carnevale di Venezia, uno degli eventi più conosciuti e celebrati d Italia. Queste maschere erano originariamente utilizzate non solo per festeggiare il Carnevale ma anche per nascondere l identità e lo status sociale di chi le indossava, consentendo così una temporanea sospensione delle norme sociali e gerarchiche. Nello specifico, porterei una maschera di Colombina, perché rappresenta l ingegno e la vivacità femminile nel cuore della Commedia dell Arte. Questa maschera, leggera e graziosa, simboleggia l astuzia e l indipendenza, celebrando il ruolo attivo delle donne nella storia teatrale. Colombina incarna la libertà di espressione e la creatività, unendo eleganza e spirito critico. Solitamente è rappresentata come una giovane donna, con lunghi capelli ricci o intrecciati, un volto vivace ed espressivo, e un abbigliamento colorato e ornamentato, spesso con una maschera bianca che copre solo gli occhi o il naso. La sua bellezza è enfatizzata da abiti eleganti e adorni, come il vestito a strati e i dettagli raffinati. Dal punto di vista artistico, ogni maschera è un opera d arte, spesso realizzata manualmente con tecniche tradizionali che si tramandano da generazioni. L artigianato richiesto per creare queste maschere è estremamente raffinato, coinvolgendo l uso di materiali come il cartapesta, il vetro e il metallo, decorati con pittura, stoffe preziose o foglia d oro. Questa attenzione al dettaglio e alla qualità riflette l elevato standard dell artigianato italiano. Culturalmente, le maschere sono un espressione dell identità veneziana e, più in generale, italiana, simboleggiando temi universali come la trasformazione e il mistero. Durante il Carnevale, Venezia si trasforma, diventando un palcoscenico dove passato e presente convivono, e dove visitatori da tutto il mondo vengono per immergersi in un atmosfera fuori dal tempo, testimoniando così l attrattiva globale della cultura italiana. Inoltre, le maschere di Venezia hanno un significato che va oltre il loro aspetto estetico e funzionale, diventando simboli di libertà e espressione individuale. Nel contesto del Carnevale, permettono alle persone di esplorare diverse identità, liberandosi temporaneamente dai vincoli sociali e personali. In conclusione, la maschera del Carnevale di Venezia rappresenta un connubio unico di storia, cultura, arte e tradizione, incarnando l essenza della creatività e dell identità italiane. Questa tradizione, ancora viva e celebrata, attira l attenzione internazionale e funge anche da ponte culturale, condividendo con il mondo intero un pezzo della ricca eredità italiana.

Label predicted: HUM (50.4%)

Explainer fit: 0.35

Figura 3.11: Versione completa del testo HUM_190_CM_F_bis[1], evidenziando la presenza delle cinque MFW selezionate. In rosso i token attribuiti alla categoria GPT e in blu quelli attribuiti a HUM.

Il testo selezionato viene correttamente classificato come prodotto da uno studente ma con una probabilità molto bassa, al limite della soglia di discriminazione. Al suo interno sono stati evidenziati i token che contribuiscono al valore finale della previsione. Dal momento che è presente sia la parola "conclusione" che aumenta il valore della variabile risposta, sia "perché" che lo diminuisce, la previsione risulta incerta. Inoltre, gli altri tre token contribuiscono a incrementare la probabilità che il testo sia stato scritto dall'intelligenza artificiale ma non a sufficienza da modificare la classe prevista.

Come evidenziato in precedenza, sebbene SHAP e LIME abbiano prodotto risultati concordanti, sono presenti alcune diversità interpretative dovute alla diversa costruzione dei due metodi. Le proprietà degli SHAP Values garantiscono che le differenze tra le singole previsioni e la previsione media nel corpus siano equamente distribuite tra i valori delle variabili esplicative relative alle unità statistiche, mentre LIME non lo assicura. Per questo motivo non è possibile affermare che, ad eccezione del metodo SHAP, i singoli contributi alle previsioni si possano sommare per ottenere il valore della previsione finale.

Un'ulteriore differenza consiste nel modo in cui LIME attribuisce i pesi nel modello semplice. LIME pesa i valori relativi alle permutazioni in base a quanto sono vicini a quelli dei dati originali, mentre SHAP attribuisce i contributi alle osservazioni campionate

[1]Il tema del testo è: "Se ti fosse chiesto di inviare una cosa che rappresenta il tuo Paese a una mostra internazionale, cosa sceglieresti? Motiva la tua posizione.".

secondo il peso che la coalizione otterrebbe nella stima del Valore di Shapley. Grazie a questo secondo modo coalizioni molto grandi e molto piccole ottengono i pesi maggiori. Questo perché si può capire molto di più sulle singole variabili se è possibile studiare i loro effetti in isolamento, e ciò avviene quando una colazione è composta da un solo predittore o quando esso è l'unico mancante.

Nel complesso, le tendenze di LIME hanno confermato le informazioni che era stato possibile ricavare grazie agli SHAP Values, rafforzando la presenza di strumenti efficaci per incrementare la trasparenza dei modelli di *machine learning* e consentendo di ottenere una comprensione più approfondita delle previsioni.

3.3.2 Analisi delle Corrispondenze

L'efficacia dell'Analisi delle Corrispondenze nel distinguere i due tipi di autore diventa evidente se si proiettano i documenti contenuti nel *test set* su un piano cartesiano, dove le coordinate dei testi sono determinate dagli assi della CA che si vuole selezionare. Gli assi di interesse sono quelli che permettono una divisione dei testi in base all'autore nel modo più chiaro e netto possibile. Un punto di partenza è rappresentato dalle prime due dimensioni (assi) che, per costruzione, possiedono i valori di varianza spiegata più elevati.

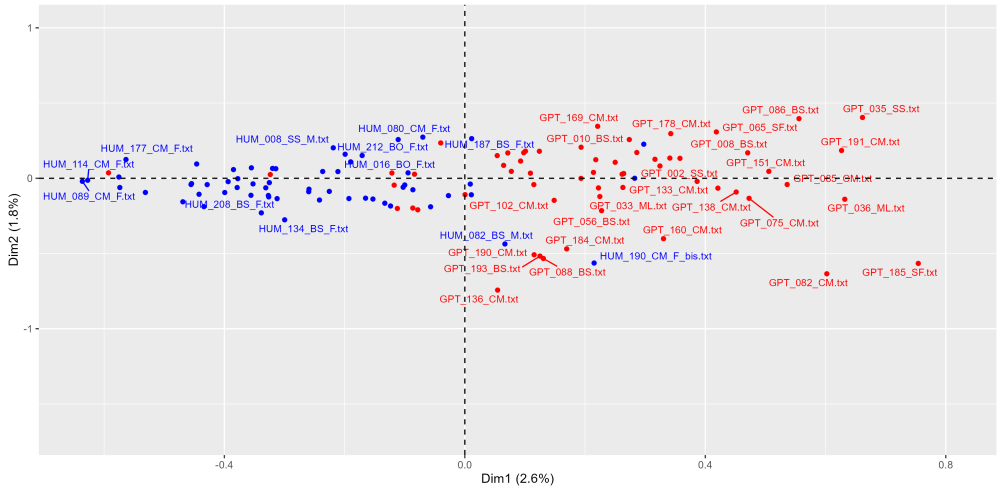


Figura 3.12: Proiezione dei testi appartenenti al *test set* in un piano cartesiano formato dalla prima e dalla seconda dimensione dell'Analisi delle Corrispondenze.

Dalla rappresentazione in Figura 3.12 risulta che la prima dimensione è in grado di isolare le due tipologie di testi in modo abbastanza chiaro. Infatti, i testi prodotti

da ChatGPT si collocano nella parte positiva dell'asse orizzontale, mentre i testi scritti dagli studenti nella parte negativa. Invece, la seconda dimensione non contribuisce in modo particolare alla divisione dei punti. Gruppi di testi che si trovano vicini nello spazio bidimensionale suggeriscono una similarità nei contenuti. In modo simile a quanto accaduto per la PCA e il MDS, si riscontra leggermente più spesso la presenza di un testo scritto da uno studente nella parte di piano appartenente ai testi prodotti artificialmente, che viceversa. Ciò costituisce un'ulteriore conferma della maggiore facilità con cui i testi della categoria GPT vengono identificati, rispetto ai testi HUM che risultano più difficili da riconoscere, anche se in questo caso il fenomeno è meno evidente.

La capacità del primo asse di isolare in modo adeguato i testi indica la sua abilità nel catturare la maggiore variabilità nei dati, come precedentemente suggerito dal suo alto valore di varianza spiegata. A tal proposito, risulta utile individuare quali sono i testi che contribuiscono maggiormente a questa dimensione. Nell'Analisi delle Corrispondenze, ogni asse rappresenta una direzione di variazione dei dati; selezionandone uno, i testi che hanno un contributo più elevato sono quelli più influenti nella sua definizione.

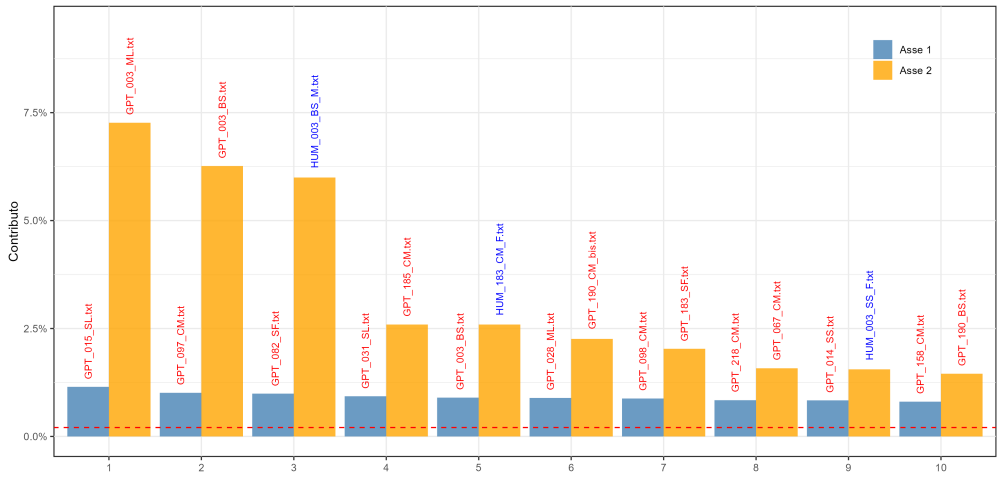


Figura 3.13: Istogramma dei testi che contribuiscono maggiormente alla definizione dei primi due assi, con indicato il contributo percentuale all'asse a cui si riferiscono. La linea rossa tratteggiata mostra il contributo atteso da ogni unità se la contribuzione fosse uniforme. Ogni valore al di sopra della linea è considerato significativo.

La Figura 3.13 mostra le righe (i testi) con l'impatto relativo più alto nella determinazione della direzione del primo e del secondo asse. Come si può vedere, i dieci maggiori contributi alla prima dimensione provengono da testi prodotti da ChatGPT, stando a significare che questo asse è dominato da questa categoria. Per quanto detto in prece-

denza, le caratteristiche di questo tipo di testi sono particolarmente differenziate dalle altre e ciò rappresenta un buon segno della capacità dell'Analisi delle Corrispondenze di distinguerle adeguatamente.

Invece, la seconda dimensione non è rappresentata da testi appartenenti ad un'unica categoria ma, anche in questo caso, i contributi provengono principalmente da testi prodotti da ChatGPT. In particolare, cattura l'attenzione la molteplice presenza del tema numero tre (riferito alla diffusione della lingua inglese) presente nei testi GPT_003_ML, HUM_003_BS_M e HUM_003_SS_F che contribuiscono al secondo asse e nel testo GPT_003_BS che contribuisce a entrambi.

Documenti che trattano lo stesso tema tendono a utilizzare un insieme simile di parole e frasi, creando una coerenza linguistica che si riflette nelle dimensioni della CA. La ripetizione di termini specifici e concetti collegati rende questi testi più simili tra loro rispetto ad altri presenti nel corpus. Siccome la CA è in grado di identificare *cluster* naturali nei dati, testi che discutono lo stesso argomento tendono a formare un gruppo distintivo quando proiettati nello spazio bidimensionale. Questo raggruppamento riflette le similitudini nei contenuti e nell'uso del linguaggio, risultando in un contributo maggiore al secondo asse, da parte di questi testi tematicamente correlati.

L'interesse si sposta dunque sull'esaminare quali caratteristiche contribuiscono all'elevata rappresentazione dei testi prodotti da ChatGPT, dal momento che essi saturano significativamente i primi due assi, in grado di isolare efficacemente gli autori.

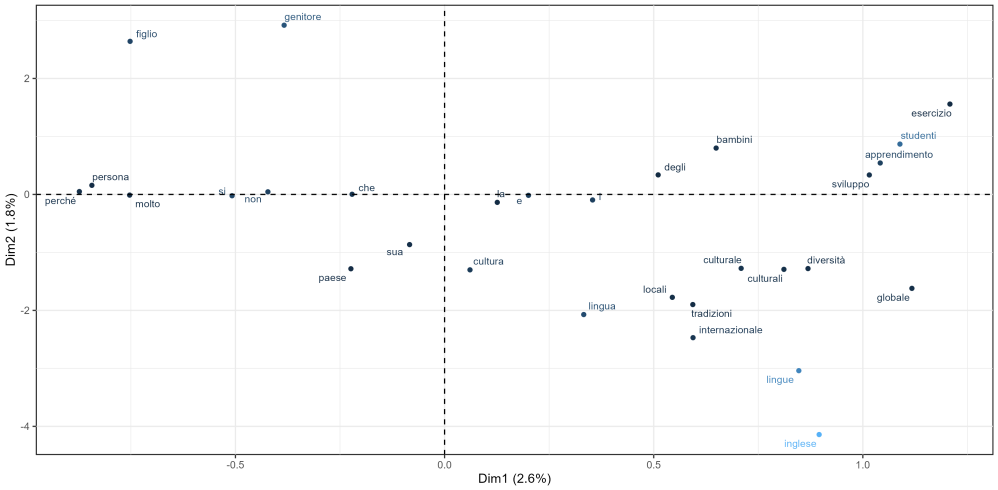


Figura 3.14: *Biplot* dei 30 termini col contributo maggiore alla varianza dei primi due assi. I termini con i valori di contributo più alti sono rappresentati con tonalità di blu più chiare.

La visualizzazione proposta in Figura 3.14 permette di identificare le parole che contribuiscono maggiormente alla costruzione dei primi due assi della CA. Vengono selezionati i primi 30 token in termini di contributo alla spiegazione della varianza lungo le prime due dimensioni. Se una parola ha un alto contributo su un asse, significa che essa è particolarmente importante nel determinare la posizione dei testi lungo quell'asse. Di conseguenza, essa costituisce una delle principali caratteristiche che differenziano i testi lungo quella dimensione.

Concentrandosi sull'asse orizzontale, le parole che lo saturano meglio si collocano alle estremità del grafico. Dalla rappresentazione emerge che i termini sono principalmente parole piene come "studenti", "apprendimento" e "sviluppo". Al contempo, al centro degli assi sono presenti alcune *stopwords* tra cui "e", "si", "che" e "non" che sono altrettanto in grado di dare un contributo elevato alla variabilità dei dati. È più difficile affermare che queste parole vuote caratterizzino gli assi ma probabilmente la loro capacità discriminatoria deriva dall'elevata frequenza con cui compaiono, che determina anche la loro posizione nel piano. L'importanza di alcuni di questi token viene confermata ulteriormente dopo l'analisi interpretativa attraverso gli SHAP Values, che aveva sottolineato l'apporto interpretativo delle *stopwords*.

Sfruttando le considerazioni precedenti si può concludere che i termini sopra citati contribuiscono alla caratterizzazione dei testi prodotti da ChatGPT e che sono in grado di saturare quasi completamente il primo asse. In aggiunta, la seconda dimensione è determinata principalmente da parole piene come "inglese", "lingue", "internazionale" e "genitore", che rispecchiano la natura del tema discusso all'interno dei testi.

L'Analisi delle Corrispondenze ha dimostrato una chiara capacità di distinguere tra testi prodotti artificialmente e quelli scritti manualmente, con una particolare efficacia del primo asse nel catturare la variabilità tra le due categorie. I testi di ChatGPT mostrano caratteristiche linguistiche distintive che permettono alla CA di isolarli efficacemente. A conferma di ciò, le parole che contribuiscono maggiormente alla costruzione delle prime due dimensioni riflettono temi e stili linguistici peculiari dei testi prodotti dal modello linguistico. Questo risultato sottolinea la potenza di questa tecnica, non solo come strumento di riduzione dimensionale, ma anche come metodo per rivelare le caratteristiche specifiche e distintive delle categorie testuali.

3.3.3 Large Language Models

Com'è noto, l'interpretazione dei risultati prodotti dai modelli linguistici pre-addestrati presenta diverse complicazioni, dovute principalmente alla loro complessità interna che

rende difficile comprendere esattamente come le decisioni vengono prese. Gli LLM utilizzano rappresentazioni ad alta dimensione delle parole e delle frasi, che catturano relazioni complesse e profonde. Sebbene queste rappresentazioni siano potenti, interpretarle in termini umani è estremamente difficile, specialmente quando si tratta di capire come diverse caratteristiche linguistiche contribuiscono alla classificazione finale. Ciò nonostante, in seguito vengono proposte due interpretazioni che possono contribuire ad incrementare la chiarezza e la trasparenza dei risultati. La prima tecnica consiste nella *Supervised Dimension Projection* (SDP), realizzata tramite la libreria `text` disponibile in `R`; mentre la seconda utilizza nuovamente gli SHAP Values ma il procedimento viene effettuato tramite il linguaggio di programmazione `Python`, per via dei suoi strumenti avanzati.

Supervised Dimension Projection

Il *Supervised Dimension Projection Plot* mostra le parole che sono significativamente correlate a un gruppo di testi rispetto a un altro, basandosi su valori di variabili considerate significative per la classificazione. Per l'analisi si è scelto di basarsi sui valori di *perplexity* e *burstiness* che hanno dimostrato di essere importanti per la previsione dell'autore, in modo da poter dividere efficacemente i testi in quattro gruppi. In sintesi, per la creazione del grafico vengono costruiti due *embedding* che catturano le differenze tra i gruppi e poi due vettori che formano due linee passanti per l'origine. Successivamente, tutti i singoli token vengono proiettati su queste linee di direzione, attraverso il calcolo dei prodotti scalari tra i vettori che identificano gli *embedding*. Più precisamente, la procedura si articola nei seguenti passaggi:
In una prima fase di *pre-processing*:

1. Le risposte (i valori di *perplexity* e *burstiness* associati ai testi) vengono divise in quattro gruppi G1, G2, G3 e G4 in base alla loro media o ai loro quartili inferiori e superiori
2. In seguito, vengono calcolati gli *embedding* aggregati delle parole presenti nei testi divisi nei quattro gruppi: l'*embedding* aggregato del gruppo G1 (in base alle parole nei testi con *perplexity* alta e *burstiness* alta), l'*embedding* aggregato del gruppo G2 (in base alle parole nei testi con *perplexity* bassa e *burstiness* bassa) e gli *embedding* relativi a G3 e G4 per le altre combinazioni dei valori. I singoli *embedding* vengono forniti in *input* assieme all'elenco dei singoli token

Al termine di questa fase vengono costruite le statistiche per il *Supervised Dimension Projection Plot*:

3. Viene definito un *embedding* relativo alla direzione aggregata per ognuno dei due assi. Per la direzione relativa alla prima variabile, l'*embedding* è calcolato come differenza tra i valori degli *embedding* G1 (alta *perplexity*) e G2 (bassa *perplexity*). Lo stesso viene fatto per la direzione relativa alla *burstiness*. Le direzioni risultanti attraversano l'origine
4. Tutti gli *embedding* delle singole parole vengono ancorati allo stesso modo. Per ciascun token l'*embedding* ancorato viene calcolato come differenza tra i valori dell'*embedding* originale e uno degli *embedding* aggregati a seconda della sua posizione di partenza
5. Per proiettare le singole parole sulle due direzioni, viene calcolato il prodotto scalare tra l'*embedding* ancorato di tutte le singole parole e l'*embedding* della direzione aggregata. I valori in ascissa e in ordinata dei punti nel grafico indicano il prodotto, ovvero la proiezione della singola parola nel grafico

Infine, si calcolano i p-value relativi alla significatività delle parole:

6. Viene creata una distribuzione nulla permutata delle proiezioni calcolando il prodotto scalare tra *embedding* di parole selezionate casualmente dai gruppi e gli *embedding* delle direzioni aggregate permutate, anch'esse create scambiando casualmente parole tra i gruppi
7. I p-value per ciascuna parola vengono calcolati confrontando la loro proiezione sulla dimensione iniziale supervisionata con la distribuzione nulla permutata, correggendo i risultati per confronti multipli

Nella rappresentazione in Figura 3.15 sono mostrate solo alcune delle parole tra quelle selezionate con un p-value basso, definite pertanto significative per la distinzione tra gruppi. Complessivamente, è possibile osservare che i punti si dispongono in tre gruppi abbastanza distinti. Il primo, associato a valori bassi di *perplexity* e *burstiness* è ben isolato; il secondo al centro del grafico rappresenta i token poco utili per la discriminazione; e infine il terzo, relativo ai testi con valori elevati di *perplexity* e *burstiness* si colloca in alto a destra.

Per i risultati raggiunti fino a questo punto si può affermare con elevata sicurezza che le parole appartenenti al primo gruppo siano frequentemente associate ai testi prodotti da ChatGPT, mentre quelle del terzo gruppo ai testi scritti dagli studenti. La disposizione dei punti nel grafico conferma la tendenza dei token appartenenti ai testi artificiali ad essere più facilmente distinguibili, rispetto alle parole tipiche dei testi umani. Non sono presenti parole significative nelle regioni del grafico relative a testi con *perplexity* alta e *burstiness* bassa, o viceversa, dal momento che queste due variabili sono correlate

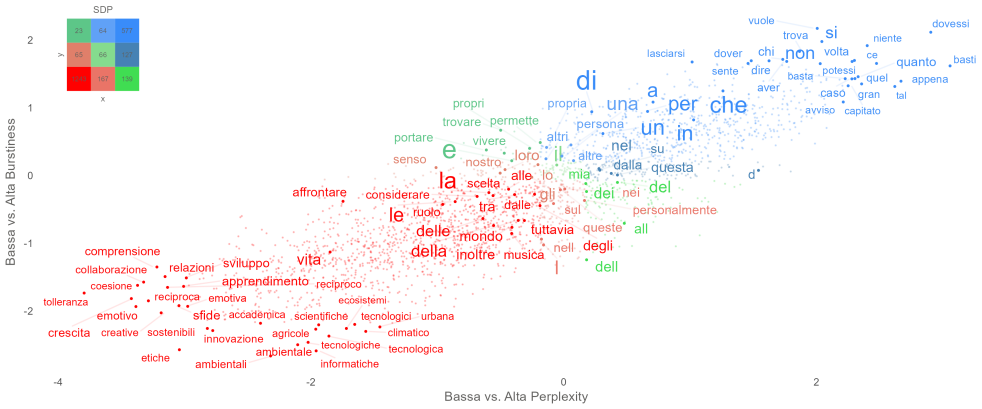


Figura 3.15: *Supervised Dimension Projection Plot* di parole significativamente differenti tra valori alti e bassi di *perplexity* (asse x) e valori alti e bassi di *burstiness* (asse y). La grandezza delle parole indica la loro frequenza. Il colore indica se una parola è significativa o non significativa (verde), dopo la correzione per confronti multipli. La legenda dei colori in alto a sinistra indica il colore e il numero di parole significative in ogni parte del grafico, selezionate tra quelle con frequenza almeno pari a 10. I punti rappresentano la posizione per ogni token, ma solo alcuni presentano un'etichetta. I valori sugli assi rappresentano il valore di proiezione in seguito al prodotto tra *embedding*.

positivamente e si muovono nella stessa direzione.

Le parole situate nella regione in basso a sinistra del grafico, rappresentate principalmente in rosso, includono termini come "comprensione", "collaborazione", "coesione", "sviluppo" e "apprendimento". Questi termini suggeriscono una focalizzazione su concetti relativi all'educazione, alla crescita personale e allo sviluppo relazionale. Il gruppo è caratterizzato da un linguaggio che sembra mirato a costruire conoscenza e competenze, spesso con un tono positivo e costruttivo. Le parole come "sostenibili", "innovazione" e "tecnologiche" indicano un'attenzione verso temi scientifici e ambientali, suggerendo che questi testi potrebbero essere orientati verso l'educazione scientifica e il progresso tecnologico. La presenza di termini come "accademica" e "ambientali" rafforza l'idea che questi testi potrebbero provenire da contesti educativi e formativi. Queste considerazioni sono coerenti con le caratteristiche dei testi artificiali discusse nel Capitolo 2.

Le parole nella parte centrale del grafico, rappresentate in più colori, includono termini di uso comune e generico come "la", "il", "della", "inoltre", "mia", "dei" e "al". Questi termini non sono specificamente associati a nessuno dei gruppi distinti e tendono ad essere parole di collegamento, *stopwords*, o parole comuni che appaiono frequentemente in molteplici contesti, come evidenziato dalla loro grandezza. La loro posizione al centro indica che

non contribuiscono significativamente alla distinzione tra i testi prodotti da ChatGPT e quelli scritti dagli studenti. Sono parole che formano la struttura base del linguaggio e che in questa analisi non hanno un peso specifico nel differenziare i due tipi di testi, ma possono comunque risultare rilevanti in altri contesti per via della loro elevata frequenza nel corpus.

Le parole in alto a sinistra, rappresentate principalmente in blu, includono termini come "per", "che", "non", "quanto", "si", "chi", "dire", "vuole" e "potessi". Questi termini indicano un linguaggio più colloquiale e discorsivo, tipico delle interazioni quotidiane e dei dialoghi. Le parole in questa categoria suggeriscono una maggiore complessità e variabilità nel linguaggio, tipico di testi che potrebbero essere più espressivi e meno formali. La frequenza di congiunzioni, preposizioni e pronomi indica un linguaggio più fluido e dinamico, che potrebbe essere più tipico dei testi scritti dagli studenti, dove c'è una maggiore varietà di espressioni e strutture linguistiche.

L'analisi delle parole nel grafico conferma la distinzione tra i testi prodotti da ChatGPT e quelli scritti dagli studenti. Le parole associate ai testi di ChatGPT tendono a essere più formali e orientate alla conoscenza, mentre le parole nei testi degli studenti mostrano una maggiore variabilità e complessità linguistica. Questo supporta l'ipotesi secondo la quale i testi artificiali sono più facilmente distinguibili rispetto a quelli umani, poiché tendono a utilizzare un linguaggio più tecnico e focalizzato, mentre i testi umani sono più variegati e meno prevedibili. Confrontando i token con quelli che si sono rivelati significativi nella *feature extraction* emerge che né "perché" né "conclusione" compaiono nel grafico. Questo può essere dovuto alla loro bassa significatività o alle procedure interne di calcolo per la rappresentazione. Al contrario, i token "si" e "non" si collocano correttamente nella porzione di grafico relativa ai testi scritti dagli studenti.

SHAP Values

Come ulteriore analisi, tramite la libreria `shap` disponibile in `Python` è possibile calcolare gli SHAP Values relativi a singole porzioni di un testo, al fine di spiegare le previsioni fornite da un modello linguistico utilizzato per la classificazione. Questo procedimento è effettuato servendosi di un LLM per dividere il testo in token e di un ulteriore modello per la classificazione delle sequenze di testo. Tramite la funzione *softmax* vengono calcolate le probabilità di appartenere a ciascuna classe e viene generato l'*output* di previsione.

Successivamente, si utilizzano gli SHAP Values per analizzare come le varie porzioni di testo hanno contribuito alla creazione della previsione finale e per ottenere l'importanza di ogni token. I valori di importanza per i modelli di testo sono spesso gerarchici e

seguono la struttura del testo. Le interazioni non lineari tra gruppi di token possono essere utilizzate durante il processo di previsione; per cui: gruppi di parole con effetti non lineari forti tra loro verranno automaticamente uniti per formare blocchi coerenti.

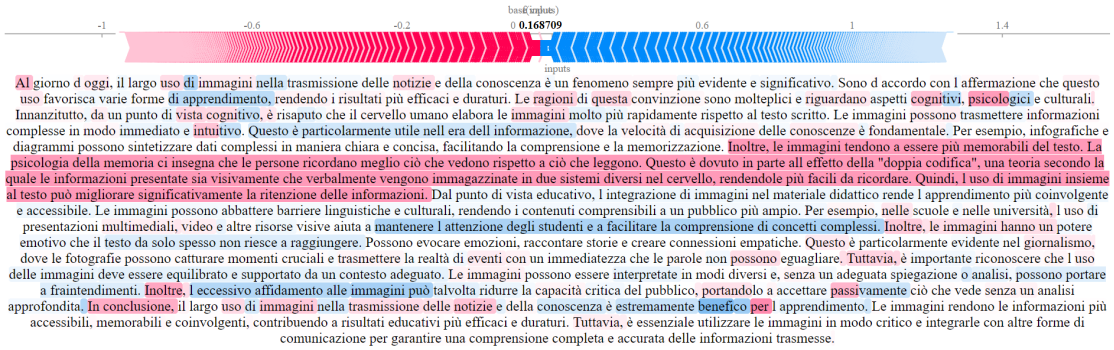


Figura 3.16: Analisi del testo GPT_001_BO[2] tramite SHAP Values per visualizzare l'importanza di ogni token. Le regioni in rosso corrispondono alle parti del testo che aumentano l'*output* del modello quando sono incluse, mentre quelle in blu lo decrementano. Colori più accesi indicano contributi maggiori in valore assoluto. Il valore della variabile risposta in grassetto rappresenta la trasformazione *logit* della previsione finale.

Per questa analisi è stato selezionato il testo GPT_001_BO appartenente alla categoria GPT e la previsione è stata effettuata nuovamente tramite *bert-base-italian-uncased*, sia per la creazione dei token, sia per la classificazione. La classe positività è stata attribuita a GPT e, di conseguenza, porzioni di testo che incrementano l'*output* propendono verso questa etichetta.

Il grafico in Figura 3.16 fornisce una panoramica di come tutte le parti del testo si combinano per formare l'*output* del modello. Essendo un testo prodotto da ChatGPT, la maggior parte delle porzioni di token incrementa la previsione. I connettivi come "in conclusione", "inoltre" e "tuttavia" emergono come parole significative comparendo anche più volte nel testo. Una parte centrale del testo viene analizzata come blocco unico e contribuisce significativamente alla classificazione. Anche in questa analisi i singoli SHAP Values possono essere raffigurati come forze contrastanti che si oppongono per decretare il valore di previsione finale. In questo caso, applicando la funzione inversa della trasformazione *logit* all'*output* del modello, si ottiene un valore pari a 0.54 per la variabile risposta. Il testo è stato dunque classificato correttamente ma con un'elevata

[2]Il tema del testo è: "Al giorno d'oggi, il largo uso di immagini nella trasmissione delle notizie e della conoscenza favorisce varie forme di apprendimento, rendendone più efficaci e duraturi i risultati". Sei d'accordo o no con questa affermazione? Motiva la tua posizione.".

incertezza, motivata dalla presenza di porzioni di testo che propendono verso direzioni opposte.

Nonostante la complessità dei modelli linguistici, le tecniche proposte hanno permesso di comprendere come sono state generate le previsioni e di identificare più precisamente quali parole o porzioni di testo hanno contribuito maggiormente alla propensione verso le due classi di risposta. Grazie a questi metodi, è stato possibile ottenere una visione più chiara delle dinamiche interne del modello e del suo processo decisionale, migliorando così la sua trasparenza e affidabilità.

Capitolo 4

Modelli di classificazione alla prova

Nel capitolo precedente è stato approfondito il processo di classificazione dei testi. Sono stati ottenuti buoni risultati che hanno permesso di affermare con elevata sicurezza che è possibile risalire alla provenienza di un documento, sia servendosi di strumenti complessi come i modelli linguistici, sia sfruttando tecniche più comuni come l'analisi delle corrispondenze o l'estrazione di indicatori e parole frequenti. È ora di interesse valutare se i risultati ottenuti possono essere estesi ad un contesto più ampio.

4.1 Confronto con i rilevatori automatici

Nel primo capitolo erano state valutate le *performance* di alcuni rilevatori automatici, selezionati tra quelli disponibili in rete. Questi strumenti misurano la percentuale di testo attribuibile all'intelligenza artificiale, all'interno di un documento che viene loro sottoposto. Attraverso metriche di valutazione create appositamente erano state testate le loro capacità su un insieme di testi in lingua inglese, nel distinguere alcuni testi completamente scritti da ChatGPT e altri completamente scritti da alcuni studenti. Questi risultati possono essere confrontati con le previsioni ottenute tramite le tecniche discusse nel Capitolo 3, per stabilire quale approccio abbia funzionato nel modo migliore.

Osservando la Tabella 4.1 è possibile eseguire un paragone tra i due approcci e valutare le eventuali analogie e differenze. I rilevatori automatici erano stati testati su 60 testi, di cui 30 per tipo, con dimensioni inferiori a 500 token. Dal momento che neanche in quel caso i testi erano stati modificati in alcun modo e in entrambe le circostanze i testi artificiali sono stati generati da ChatGPT-3.5, il confronto con i testi in italiano è legittimo.

Tabella 4.1: Confronto tra i migliori risultati prodotti nel Capitolo 1 e nel Capitolo 3. *Feature Extraction* fa riferimento alla tipologia `Text_Features_svmRadial_LGOCV`, Analisi delle Corrispondenze fa riferimento a `CA-15_svmRadial_repeatedcv` e *Large Language Models* fa riferimento a `Bert-ita-15_svmRadial_repeatedcv`. I modelli sono ordinati per Accuratezza decrescente, evidenziata in **grassetto**.

Strumento	Accuratezza	Errore totale	Precision	Recall	Specificity	F1
Feature Extraction	**1**	0	1	1	1	1
Analisi delle Corrispondenze	**1**	0	1	1	1	1
Large Language Models	**0.975**	0.025	0.952	1	0.950	0.976
Contentatscale ai detector	**0.967**	0.033	0.966	0.967	0.967	0.967
QuillBot	**0.950**	0.050	1	0.967	0.933	0.983
GPT-2 Output Detector Demo	**0.933**	0.060	0.906	0.967	0.900	0.935
Kazanseo detector	**0.933**	0.067	1	0.933	0.933	0.965
ZeroGPT	**0.933**	0.067	1	0.933	0.933	0.965

Come precedentemente dichiarato, dall'utilizzo dei rilevatori automatici è emersa la loro elevata capacità di riconoscere i testi scritti dall'intelligenza artificiale. Questa affermazione è sostenuta dagli alti valori di *Precision* e *Recall* (*Sensitivity*). Inoltre, anche i valori di *Specificity* sono particolarmente elevati, ma leggermente inferiori ai precedenti. D'altra parte, i risultati ottenuti attraverso le tecniche sviluppate in questa tesi sono superiori. In molti casi è stato raggiunto un livello di Accuratezza pari a 1, rispetto al valore di 0.97 ottenuto dal rilevatore automatico più efficiente.

Le metriche proposte hanno messo in luce l'abilità dei rilevatori automatici nel riconoscere con più sicurezza un testo prodotto da ChatGPT rispetto ad uno scritto da uno studente. Questo aspetto è comparso più volte nel corso di questo lavoro e costituisce una delle considerazioni maggiormente consolidate dai risultati ottenuti: i testi artificiali sono più prevedibili e vengono identificati abbastanza facilmente se disposti in un piano bidimensionale, anche utilizzando criteri e tecniche differenti.

Nel Capitolo 2 è stato mostrato come bastassero le prime dimensioni della PCA basata sulle MFW per separare quasi completamente le due tipologie di testi. Se venivano utilizzate le Text Features o si univano gli approcci, i risultati erano ancora migliori. Tecniche come il MDS o la *Cluster Analysis* hanno condotto a risultati simili. Nel Capi-

tolo 3 la rappresentazione dei testi nel piano creato dai primi due assi dell'Analisi delle Corrispondenze ha confermato queste ipotesi: i documenti generati da ChatGPT sono isolati agevolmente, disponendosi nella stessa parte di piano, mostrando la loro elevata similarità. In aggiunta, attraverso il *Supervised Dimension Projection Plot* prodotto grazie all'utilizzo di un modello linguistico, è stato possibile risalire a quali termini rendessero riconoscibile questa tipologia di testi in maniera abbastanza evidente. Anche osservando il primo asse della CA sono emersi i token associati frequentemente ai testi artificiali e il confronto con quanto ottenuto con le altre tecniche è piuttosto consistente.

Come ribadito, questo accade perché ChatGPT è basato su architetture neurali che producono testo con una struttura, stile e uso del linguaggio altamente coerenti. Questa peculiarità rende i testi così generati più facilmente riconoscibili rispetto ai testi umani, che tendono a variare di più in termini di stile e struttura. I modelli linguistici seguono sequenze specifiche nella costruzione delle frasi e nell'uso delle parole e possono mostrare una certa ripetitività nei temi trattati, così come nelle formulazioni effettuate. Queste caratteristiche sono in grado di creare dei segnali distintivi facilmente catturabili dai modelli di *machine learning* utilizzati per la classificazione. Per questo motivo sono state usate metriche come *perplexity*, *burstiness*, `sd_tok_per_frase` e `sd_lunghezza_token` che cercassero di individuare questa ripetitività e prevedibilità. Infatti, molte di queste variabili si sono rivelate le più importanti per la loro capacità predittiva.

Al contrario, nel caso dei testi umani, si è notato come avessero spesso delle caratteristiche simili e distintive che contribuivano a renderli isolabili dai modelli di classificazione, ma che al contempo, alcuni di loro si confondessero tra i testi prodotti da ChatGPT, provocando previsioni errate. A dimostrazione di ciò si possono osservare le disposizioni dei testi umani nello spazio delle componenti principali o del MDS. Inoltre, l'Analisi delle Corrispondenze ha confermato questa tendenza grazie alle sue proiezioni in uno spazio a bassa dimensionalità dove le distanze riflettono somiglianze. Come si è visto, i testi generati da ChatGPT, a causa della loro coerenza stilistica e dei pattern linguistici, tendono a raggrupparsi in *cluster* distinti, mentre per i testi umani è più difficile che ciò accada. Gli scritti umani variano notevolmente in termini di stile, argomento, tono, e complessità. Questa varietà si riflette in una distribuzione più dispersa nello spazio di rappresentazione, rendendo i testi umani meno omogenei e più diffusi.

Tuttavia, l'originalità e la creatività umana possono portare a stili di scrittura che sono nettamente diversi tra loro, rendendo anche questi testi difficilmente identificabili. Testi umani che sono scritti con un linguaggio molto generico o privo di peculiarità stilistiche distintive possono essere confusi con i testi generati da ChatGPT. Questi testi mancano delle caratteristiche uniche che potrebbero altrimenti aiutare i modelli di clas-

sificazione a isolarli correttamente. La complessità e la varietà intrinseca dei testi umani possono superare le capacità di generalizzazione dei modelli di *machine learning*. Anche le tecniche più avanzate possono commettere errori di previsione quando incontrano testi umani che non seguono gli schemi comuni o presentano ambiguità.

Per questi motivi, mentre molti testi umani possono essere isolabili dai modelli di classificazione grazie alle loro qualità distintive, esistono comunque documenti che, per la loro semplicità o per somiglianze stilistiche con i testi generati dall'intelligenza artificiale, sono in grado di confondere i modelli, risultando in previsioni errate. Infatti, tra i modelli che non sono stati capaci di classificare correttamente l'intero *test set*, quasi tutte le classificazioni errate rientrano nella tipologia di Falsi Positivi, ovvero testi scritti da studenti a cui viene scorrettamente attribuita l'etichetta GPT, come peraltro accade con i rilevatori automatici analizzati.

Nel Capitolo 1 era stato affermato che l'utilizzo combinato di diversi rilevatori avrebbe potuto aumentare la robustezza dei risultati, incrementando la precisione delle previsioni. Si può affermare che utilizzare diversi *set* di *feature* o tecniche differenti abbia confermato questa ipotesi. L'utilizzo di strumenti come MFW, n-grammi, indicatori testuali, caratteristiche semantiche ed *embedding* ha permesso di catturare un raggio più ampio di informazioni sui testi. Ogni quantità ha consentito di evidenziare aspetti diversi del linguaggio, migliorando la capacità di discriminazione dei modelli. Inoltre, questa pratica ha permesso di evitare l'*overfitting*, riducendo il rischio che il modello si adattasse troppo ai pattern specifici dei singoli testi e migliorando la generalizzabilità. Anche la riduzione della dimensionalità ha contribuito a conservare solamente le informazioni più significative, mettendo in evidenza le caratteristiche rilevanti. L'aspetto di combinare rilevatori differenti assume ancora più significato se si considera la scelta dei modelli di *machine learning* utilizzati: servirsi di tipologie differenti ha reso possibile sfruttare i punti di forza di ciascuno di essi (come l'abilità nel catturare relazioni non lineari o nel gestire frequenze particolarmente differenti tra loro). In aggiunta, tecniche come il *bagging* e il *boosting* hanno consentito di combinare le previsioni di modelli deboli per ottenere un risultato più robusto.

Ciò nonostante, in seguito a queste considerazioni non è possibile dichiarare con tranquillità che i modelli costruiti in questo lavoro siano in grado di rilevare più efficacemente la presenza di testo prodotto dall'intelligenza artificiale, rispetto ai rilevatori automatici disponibili in rete. Sebbene sia vero che questi ultimi abbiano prodotto risultati peggiori in un compito di classificazione simile, bisogna anche considerare che sono stati creati per funzionare in qualsiasi linguaggio e per molteplici tipologie di testi. Probabilmente, facendo un banale confronto in un'altra lingua o con testi strutturati diversamente, essi

avrebbero i risultati migliori. Le tecniche discusse in questa tesi sono state sviluppate e perfezionate per lavorare su una specifica tipologia di documenti, con determinate caratteristiche e uno stile definito. Anche la creazione dei rilevatori automatici è avvenuta in maniera simile, ma estesa per un utilizzo più ampio. La fase di *fine-tuning* di GPTZero si è basata su dati che includono una gran numero di materiale prodotto da studenti, in modo da aumentare l'accuratezza del rilevamento per scopi educativi; inoltre, la fase di *pre-training* di Open AI classifier è stata effettuata adoperando coppie di testi scritti da esseri umani e dall'intelligenza artificiale sullo stesso argomento. Questo dimostra che l'approccio di base è il medesimo ma l'addestramento dei classificatori è molto più esteso e generalizzato.

4.2 Modifica dei testi

In un'ottica di possibile applicazione pratica per le tecniche sviluppate in questa tesi, risulta facile pensare che difficilmente sarà richiesto di discriminare testi scritti interamente da persone e testi completamente generati da un modello linguistico. Come anticipato, un possibile utilizzo per questi strumenti può avvenire in ambito educativo, dove vi è la necessità di verificare l'autenticità di un compito prodotto da uno studente o la legittimità di un lavoro svolto in autonomia. In queste situazioni, un alunno che decidesse di servirsi di un modello linguistico per generare del testo, è probabile che esegua dei ritocchi per rendere meno evidente la sua natura. La modifica può avvenire tramite la sostituzione di porzioni di testo troppo ricercate o elaborate o semplicemente utilizzando alcuni sinonimi per parole che difficilmente si possono trovare in testi scritti da ragazzi.

In questa fase si vuole dunque mettere alla prova i modelli e le tecniche sviluppate in questo lavoro, andando ad apportare modifiche mirate ai testi, per osservare come cambiano le probabilità di attribuzione dei documenti ai due autori. Lo scopo di questa operazione non è solo dare un senso pratico a quanto fatto fino a questo punto, ma anche verificare la robustezza dei metodi elaborati e osservare l'andamento delle variabili risposta in seguito a modifiche progressive ai valori delle variabili esplicative su cui si basano.

Per fare ciò viene selezionato un insieme di dieci testi appartenenti alla categoria GPT, che sono stati classificati in modo corretto con elevata sicurezza. Per via della natura delle modifiche che verranno effettuate si sceglie di basarsi sulle previsioni effettuate dalla combinazione `Text_Features+MFW_xgbTree_repeatedcv`: un modello che viene addestrato su caratteristiche testuali basate sia sugli indicatori linguistici, sia sulle parole più frequenti. I testi scelti hanno una probabilità di appartenere alla categoria

GPT che varia da 0.94 a 1, stando ad indicare le loro evidenti caratteristiche di artificialità. Scegliendo testi così ben riconoscibili è possibile osservare l'effetto delle modifiche su documenti che partono da un livello di elevata identificabilità e come esso vari progressivamente. Si vuole dunque alterare manualmente questi testi per poi applicare lo stesso modello predittivo usato sui testi originali, per vedere se e come si modifica la variabile risposta (che indica la probabilità di appartenenza alla classe GPT) associata ad essi.

Per la scelta di quali cambiamenti effettuare sui testi ci si è basati sull'importanza dei predittori per il modello, calcolata con il metodo `varImp` di `caret`, i cui valori sono stati riportati in Figura 3.3. Osservando l'elenco di variabili per il modello scelto era emerso che i predittori più importanti fossero *perplexity*, `sd_tok_per_frase` e *burstiness* come indicatori testuali e "perché", "conclusione", "non" e "si" come parole frequenti. In realtà la capacità discriminatoria di queste variabili si è manifestata più volte nello studio delle varie tecniche e quindi si può affermare che il cambiamento dei valori dei predittori possa avere un impatto significativo sugli *output* del modello.
Per i motivi sopra elencati, sono state operate le seguenti modifiche ai testi selezionati:

1. Rimozione del token "conclusione" nei casi in cui è presente
2. Aggiunta del token "perché"
3. Aggiunta dei token "si" e "non"
4. Accorpamento di frasi distinte in un'unica frase
5. Modifiche specifiche in parti complesse, uso di sinonimi e riformulazione di frasi

La rimozione del token "conclusione" per rendere i testi più umani è motivata dallo studio tramite SHAP Values effettuato nel Capitolo 3; la stessa cosa vale per i token "si" e "non", la cui elevata presenza aiuta i testi ad essere classificati come scritti dagli studenti. Inoltre, diminuendo il numero di frasi si vanno a creare testi meno frammentati e più discorsivi, tipici delle produzioni umane. Così facendo, non solo varia il numero di frasi e di token per frase ma anche la loro dispersione, visto che solo alcune delle frasi vengono mutate. Infine, lo scopo della quinta e ultima modifica è rendere i testi meno riconoscibili dai modelli, sostituendo le parti ritenute più facilmente identificabili e cercando di aumentare l'imprevedibilità che nei testi generati artificialmente solitamente manca. L'elenco delle modifiche effettuate e la variazione dei valori per le frequenze relative e per gli indicatori linguistici possono essere esaminati nella Tabella 4.2.

Tramite i valori si può notare che il token "conclusione" è inizialmente presente in quasi tutti i testi selezionati; infatti, ha una frequenza media di poco inferiore ad 1. Dal momento che ad ogni passo viene modificato il testo creato nella fase precedente, le frequenze di questa parola rimangono pari a 0 dalla prima modifica in poi. Al contrario,

Tabella 4.2: Medie, calcolate sui 10 testi, dei valori per ogni *feature* e delle frequenze assolute per i token nelle varie fasi del processo di modifica dei testi.

	Val. iniziali	**Mod. 1**	**Mod. 2**	**Mod. 3**	**Mod. 4**	**Mod. 5**
conclusione	0.8	0	0	0	0	0
perché	0.3	0.3	4.1	4.1	4.1	4.1
si	2.6	2.6	2.6	6.4	6.4	6.3
non	2.4	2.4	2.4	7.3	7.3	7.2
`sd_tok_per_frase`	6.592	6.566	6.586	6.689	13.893	13.777
perplexity	133.240	135.432	146.273	157.999	167.756	169.513
burstiness	10.637	10.678	10.615	10.545	10.550	10.569

nel secondo punto viene ampliata la presenza del token "perché" nei testi, facendo alzare la frequenza media assoluta di quasi quattro unità. In modo analogo, per la modifica successiva viene aggiunto circa quattro volte il token "si" ai testi e quasi cinque volte il token "non", incrementando notevolmente le loro frequenze assolute medie.

Per quanto riguarda i cambiamenti volti a ridurre la ripetitività e la prevedibilità dei testi, con la quarta modifica viene fatta variare maggiormente la lunghezza delle frasi, che comporta un notevole aumento della variabile `sd_tok_per_frase`. Va notato però che anche le tre modifiche precedenti hanno un leggero impatto su questo predittore, come si può notare dai suoi valori nella tabella. Infine, l'ultima modifica è volta a incrementare i valori di *perplexity* e *burstiness* nei testi. Eppure, sebbene questo avvenga se si considera solo il passaggio dal penultimo *step* all'ultimo, ciò non sempre accade se si prendono i valori nella loro totalità.

Come si può notare, i valori di *perplexity* sono sempre crescenti nel corso delle variazioni ai testi; tuttavia, le prime quattro modifiche hanno gli effetti desiderati ma, al contempo, hanno lo svantaggio di ridurre leggermente i valori di *burstiness*. Questo avviene per come è calcolato l'indicatore: l'aggiunta dei token "perché", "si" e "non" fa aumentare il suo numeratore ma contemporaneamente decrescere il denominatore, relativo alle frequenze medie dei token, con il risultato che la frazione complessiva risulta inferiore. Questo effetto è il contrario di quanto desiderato, anche se nell'ultima modifica i valori di *burstiness* effettivamente aumentano mediamente. Inoltre, effettuando quest'ultimo cambiamento, le frequenze assolute di "si" e "non" decrescono leggermente e anche il valore medio di `sd_tok_per_frase` diminuisce di poco.

L'aspetto però che è da tenere maggiormente in considerazione è che modificando i testi, non variano solo le sei variabili esplicative prese in considerazione, ma l'intero insieme di Text Features e le frequenze di molte MFW usate come predittori. Infatti,

rendendo maggiore il numero di frasi variano anche quantità come "frasi", `tok_per_frase`, `caratteri_per_frase` e molti altri. Senza considerare che modificando anche solo una piccola parte di testo può cambiare il numero totale di nomi, aggettivi, pronomi e così via.

Per questo motivo si è deciso di procedere in due modi differenti: come prima analisi viene studiata la variazione delle probabilità di attribuzione dei testi alla categoria GPT, sostituendo ai predittori solamente i valori delle variabili esplicative elencate in precedenza. Successivamente, si ripete l'analisi ricalcolando completamente tutti i valori delle variabili, nei casi in cui essi sono mutati. Questo secondo processo è più realistico, se si vuole trovare un risvolto pratico alla previsione dell'autore, ma ha lo svantaggio di non poter isolare l'effetto dei singoli cambiamenti nelle previsioni.

Nell'analisi non è stato considerato il procedimento opposto, ovvero la modifica di testi umani da parte di ChatGPT, principalmente per la poca utilità che si ritiene abbia questo cambiamento. Ci si aspetta che, ad esempio a livello scolastico, sia più diffuso l'utilizzo di modelli linguistici per generare testo, piuttosto che per sostituire ciò che si è già scritto; anche se una pratica comune è adoperare un LLM per correggere dei documenti al fine di renderli migliori. Inoltre, come già affermato, i testi artificiali si distinguono per la loro facile individuazione e il loro isolamento rispetto all'altra tipologia.

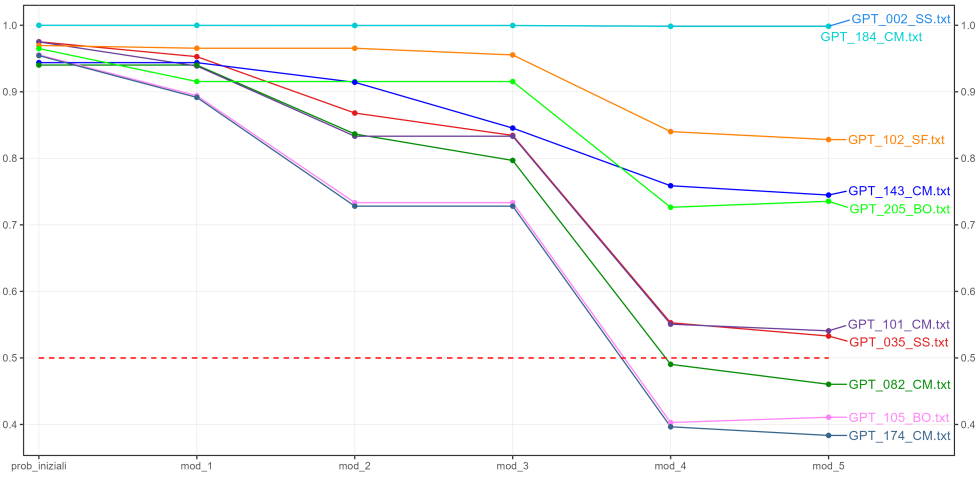


Figura 4.1: Andamento delle probabilità dei testi di appartenere alla classe GPT, in seguito alle modifiche applicate ad essi, cambiando solo i valori delle variabili presenti nella Tabella 4.2. La linea rossa tratteggiata indica il livello pari a 0.5 che pone la soglia al di sotto della quale il testo viene classificato come Human.

La Figura 4.1 mostra le previsioni del modello di classificazione applicato ai testi mo-

dificati. L'interesse ricade sulla probabilità di appartenere alla classe GPT, che quindi può variare tra 0 e 1; inoltre, come precedentemente affermato, i valori iniziali sono tutti prossimi ad 1. Da un primo sguardo d'insieme si evince che, al termine del processo, tre testi su dieci presentano una probabilità inferiore a 0.5 di essere stati scritti dal modello linguistico e quindi vengono classificati in modo errato. Andando più nel dettaglio, analizzando come variano le probabilità nel corso delle modifiche, si può affermare che, come da aspettative, la maggior parte dei testi presenta un valore inferiore a quello di partenza. Questo effetto non sorprende; tuttavia, due testi risultano comunque perfettamente classificati come prodotti da ChatGPT.

Se sia un risultato positivo o negativo dipende dall'interpretazione che si vuole dare all'analisi. Da un lato, va riconosciuta una robustezza e una solidità nel classificatore che riconosce correttamente il testo frutto dell'intelligenza artificiale, visto che un ridotto insieme di modifiche non fanno sì che si possa reputare di origine mista o scritto da uno studente. D'altra parte, è inaspettato che un cambiamento abbastanza consistente delle variabili più importanti per la classificazione non generi un naturale decremento della variabile risposta.

Gli altri testi considerati hanno invece comportamenti più prevedibili: quasi ogni modifica si tramuta in un abbassamento del valore delle previsioni, in modo diverso per le singole osservazioni. Analizzando solo le classi finali attribuite ai testi, considerato che idealmente dovrebbero essere tutti classificati con etichetta GPT ma con probabilità inferiore ad 1, si può dire che l'analisi abbia portato, in parte, i risultati sperati. I modelli si dimostrano efficaci e capaci di riconoscere la natura di un testo, anche in seguito a modifiche poco estese nella forma, ma abbastanza consistenti per l'impatto che dovrebbero avere sulla classificazione.

Passando invece alle previsioni basate sul ricalcolo completo delle variabili esplicative presenti nel modello di classificazione, i risultati cambiano secondo vari aspetti. In questo caso sono quattro i testi a cui viene attribuita l'etichetta HUM, due dei quali presenti anche nel primo caso. In generale, i testi con le probabilità più alte hanno valori elevati di previsione anche nel grafico precedente, e lo stesso vale per i testi con i valori più bassi. Inoltre, i testi `GPT_002_SS` e `GPT_184_CM` conservano le loro caratteristiche in grado di renderli altamente classificabili come prodotti da un modello linguistico, come osservabile in Figura 4.2.

Tuttavia, l'andamento dei valori nel corso delle modifiche non è più decrescente: in particolare, per alcuni testi l'aggiunta dei token "si" e "non" comporta un aumento delle probabilità e lo stesso vale per l'ultimo cambiamento apportato. Aver considerato le variazioni di tutti i predittori del modello ha influito sugli *output* ma senza stravolgere

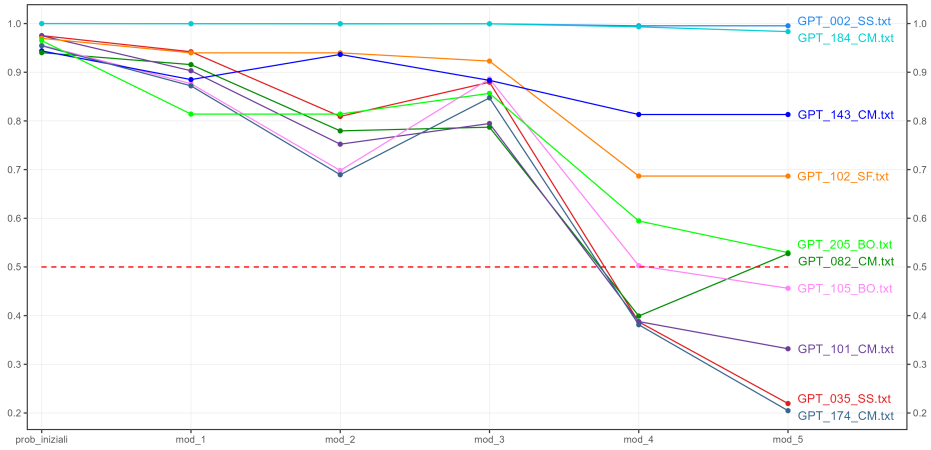


Figura 4.2: Andamento delle probabilità dei testi di appartenere alla classe GPT, in seguito alle modifiche applicate ad essi, cambiando i valori di tutte le variabili esplicative. La linea rossa tratteggiata indica il livello pari a 0.5 che pone la soglia al di sotto della quale il testo viene classificato come Human.

la loro natura. Probabilmente, gli andamenti anomali presenti per qualche testo sono dovuti agli effetti su variabili legate ad aspetti non considerati direttamente in questo processo, che hanno provocato un aumento imprevisto dei valori di previsione.

Per avere un'idea più chiara dell'impatto di ogni modifica sui valori delle previsioni del modello, è possibile analizzare le variazioni medie tra le variabili risposta in seguito ad ogni cambiamento apportato ai testi, come rappresentato in Figura 4.3.

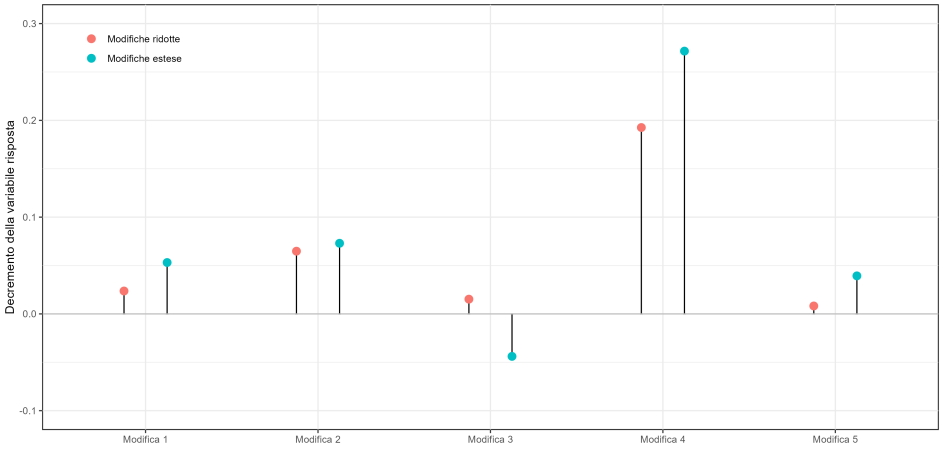


Figura 4.3: Valori medi, per i dieci testi, delle differenze tra le variabili risposta ad uno *step* del processo e al passo successivo. Vengono riportati i valori nel caso in cui si considerano le modifiche solo per un *set* ridotto di variabili (modifiche ridotte) e per il caso esteso a tutti i predittori (modifiche estese).

Come anticipato, quasi tutte le modifiche hanno un effetto diminutivo sulla varia-

bile risposta, che genera un aumento della probabilità che il testo sia classificato come HUM. Ciò non accade però per la Modifica 3 nel caso esteso, che contribuisce in media a rendere i testi più simili a quelli della categoria GPT. Come accennato, questo effetto indesiderato può essere dovuto all'aumento della ripetitività dei token che impatta su altre variabili esplicative, modificandone i valori. In generale, l'accorpamento delle frasi dei testi tramite la Modifica 4 contribuisce maggiormente alla variazione dei valori di previsione, soprattutto nel caso esteso. Questo avviene perché il cambiamento si estende a più variabili esplicative, come quelle relative al numero di frasi, numero medio e deviazione standard dei valori di token per frase, caratteri per frasi e lunghezza massima di una frase, che hanno tutte una elevata importanza. L'ultima modifica è invece quella meno significativa, per i motivi discussi in precedenza: il leggero aumento dei valori di *burstiness* e *perplexity* non riesce ad avere un buon impatto, dal momento che esso viene anche contrastato dalla diminuzione dei valori di `sd_tok_per_frase` e delle frequenze di "si" e "non".

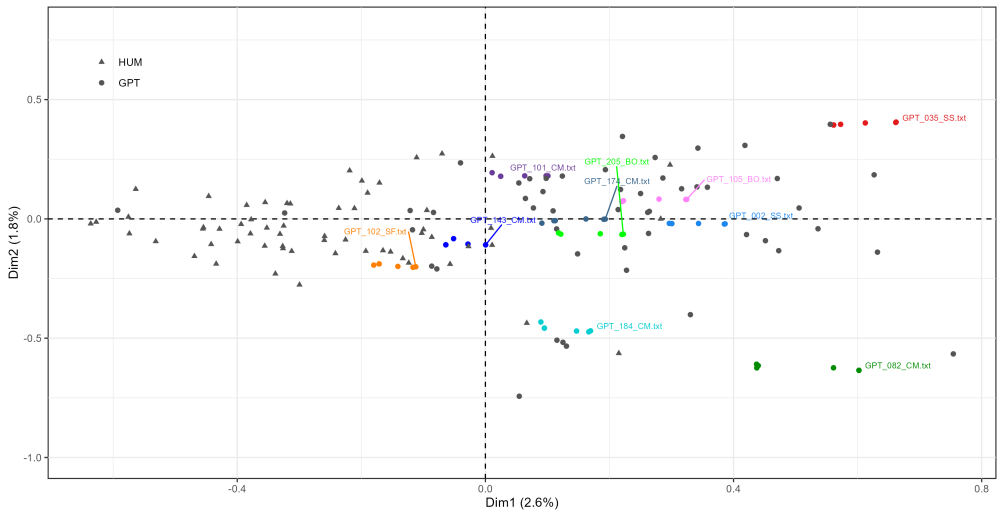


Figura 4.4: Proiezione dei testi appartenenti al *test set* e dei testi modificati in un piano cartesiano formato dalla prima e dalla seconda dimensione dell'Analisi delle Corrispondenze. Vengono colorati i punti relativi ai testi modificati e viene posta un'etichetta sui testi originali per indicare la loro posizione.

Si può ottenere un'ulteriore interpretazione delle modifiche effettuate, proiettando i testi modificati nello spazio dell'Analisi delle Corrispondenze come righe supplementari (Figura 4.4). Dal momento che la CA si basa sulle frequenze relative delle parole con almeno 50 occorrenze nel corpus, sono stati valutati i cambiamenti estesi sulle variabili, che comprendono dunque tutte le modifiche apportate alle MFW considerate, e non solo

quelle relative ai token “conclusione”, “perché”, “si” e “non”. Le modifiche sono state pensate per avere un impatto maggiore nella tecnica della *feature extraction*, ma anche in questo caso si possono notare gli effetti. I testi modificati tendono a spostarsi verso la parte di piano associata ai testi scritti dagli studenti e questo conferma la loro tendenza a divenire meno identificabili dal modello. Il grafico non mostra un’elevata analogia tra la disposizione dei punti e l’ordinamento dei testi effettuato attraverso i valori delle probabilità presenti nella Figura 4.2, ma questo è dovuto alle differenze tra le due tecniche e alle variabili su cui sono calcolate.

Il processo di modifica dei testi ha confermato che interventi mirati possono ridurre la riconoscibilità dei testi generati artificialmente, avvicinandoli maggiormente ai testi umani. Tuttavia, appare evidente che l’efficacia di tali modifiche debba essere valutata in un contesto più ampio, considerando l’interazione tra diverse variabili testuali. A questo proposito è emersa una differenza significativa nella facilità di alterazione dei valori per variabili come “conclusione” e “perché” rispetto a indicatori più complessi come *perplexity* e *burstiness*, al fine di ingannare il classificatore. Ciò nonostante, i modelli hanno risposto ai cambiamenti come previsto, dimostrando la validità degli approcci adottati e l’influenza delle tecniche di modifica applicate.

4.3 Lo sviluppo dell’Intelligenza Artificiale: un problema o un’opportunità?

Nel corso di questa tesi, è emerso come distinguere tra testi scritti da studenti e testi generati dall’intelligenza artificiale sia attualmente possibile attraverso mirati metodi statistici di classificazione. Tuttavia, è altrettanto evidente che sia sufficiente apportare modifiche minime ai testi generati artificialmente per renderli irriconoscibili. Questo suggerisce che, in un futuro non troppo lontano, potrebbe diventare estremamente difficile, se non impossibile, distinguere i testi grezzi prodotti da un modello linguistico da quelli scritti da esseri umani.

L’avanzamento dell’intelligenza artificiale solleva una serie di questioni etiche e pratiche. Da un lato, la capacità di generare testi indistinguibili da quelli umani può essere vista come una minaccia. C’è il rischio che la diffusione di contenuti generati artificialmente senza trasparenza possa minare la fiducia nelle comunicazioni scritte e facilitare la disinformazione. Questo porta alla domanda: l’intelligenza artificiale ci fa paura? La risposta non è semplice. Se da un lato esistono timori legittimi, dall’altro lato ci sono anche opportunità significative.

I modelli linguistici rappresentano un potente strumento in grado di abbattere barriere linguistiche, migliorare l'accesso all'informazione e ridurre significativamente i tempi di produzione dei contenuti. Tuttavia, è fondamentale non abusarne. È imperativo che l'uso dell'intelligenza artificiale nella generazione di testi sia dichiarato e trasparente. Questo garantisce che i lettori possano valutare criticamente le informazioni che ricevono, mantenendo intatta la fiducia nelle comunicazioni digitali.

Un altro aspetto cruciale riguarda l'abilità nel riconoscere l'impronta dell'intelligenza artificiale. Man mano che le capacità generative dei modelli linguistici continuano a progredire sarà necessario sviluppare strumenti altrettanto avanzati per identificarli. La sfida sarà mantenere il passo con l'innovazione tecnologica, evitando che le tecniche di riconoscimento rimangano indietro.

In sintesi, mentre l'intelligenza artificiale offre vantaggi significativi, il suo utilizzo deve essere governato da principi etici chiari. La trasparenza è fondamentale: dichiarare l'uso di modelli linguistici nella creazione di contenuti è essenziale per mantenere la fiducia e l'integrità delle comunicazioni. Solo attraverso un uso etico e responsabile di questi strumenti è plausibile sperare di trarre pieno vantaggio dalle loro capacità, minimizzando al contempo i rischi associati. Non è da escludere che questo stesso paragrafo sia stato generato, almeno parzialmente, con l'aiuto di ChatGPT.

Conclusioni

La rilevazione dei testi scritti dagli studenti universitari e delle corrispettive produzioni linguistiche generate da ChatGPT ha consentito lo studio di due differenti corpus, al fine di verificare se i documenti in essi contenuti fossero effettivamente distinguibili quando non è noto il loro autore.

In seguito ad una breve ma precisa panoramica sul tema in questione, le due raccolte di testi sono state esaminate accuratamente, dapprima focalizzandosi sui loro aspetti d'insieme e successivamente addentrandosi nelle peculiarità delle loro composizioni. A questo scopo si è fatto uso di tecniche note in letteratura come l'Analisi delle Componenti Principali (e.g. Wold *et al.*, 1987), il *Multidimensional Scaling* (e.g. Borg e Groenen 2005) o la *Cluster Analysis*, ma anche di strumenti più ricercati come dizionari ontologici e *tagger* semantici.

Inoltre, sono state calcolate quantità, come la *perplexity*, capaci di ispezionare la vera natura dei testi e altre maggiormente legate alle tipologie di termini presenti in essi. Queste prime analisi esplorative hanno permesso di proiettare le unità statistiche su spazi a ridotta dimensionalità, facendo emergere le loro differenze e mostrando come bastino pochi componenti per separarle in raggruppamenti quasi perfetti.

Per raggiungere risultati ottimali è stato però necessario avvalersi di tecniche di vettorizzazione capaci di trasformare il testo grezzo in rappresentazioni numeriche assimilabili dai modelli di *machine learning*. Tra i metodi adottati, il più innovativo consiste nell'utilizzo di modelli linguistici per convertire i testi in coordinate geometriche che, a seconda della loro distanza nello spazio, riflettono la somiglianza tra le rappresentazioni. I vettori numerici così creati chiamati *embedding*, affiancati ai frutti di tecniche più consolidate come l'Analisi delle Corrispondenze (Greenacre, 1984) e la *feature extraction*, hanno permesso un confronto approfondito ed esaustivo, alla ricerca del metodo migliore per discriminare i testi.

Attraverso la selezione accurata dei modelli, la calibrazione dei parametri e il controllo dell'addestramento sono stati raggiunti risultati ideali che hanno consentito di stilare un

ordinamento tra le metodologie in gioco, in base alla loro accuratezza nella classificazione. *Feature extraction* e Analisi delle Corrispondenze sono state in grado di assegnare l'autore dei testi in modo corretto per l'intero *test set* di dati a disposizione. Invece, nonostante la loro complessità e capacità di individuare relazioni profonde, l'uso dei modelli linguistici ha prodotto esiti peggiori, raggiungendo comunque buoni livelli di distinzione che tuttavia non bastano a compensare l'elevato costo computazionale richiesto dal loro utilizzo.

Tuttavia, anche le tecniche più efficienti hanno poca utilità se non si è in grado di interpretare i loro risultati. Fortunatamente, tramite strumenti come SHAP (Lundberg e Lee, 2017) e LIME (Ribeiro *et al.*, 2016) è stato possibile risalire alle *feature* più rilevanti nella classificazione e soprattutto al loro contributo. Come era prevedibile alcuni token, come "conclusione", vengono individuati più frequentemente nei documenti della classe GPT ed altri, ad esempio "perché", sono presenti maggiormente nell'altra tipologia di testi. Più rilevante è stato osservare come le variabili dei modelli si influenzassero a vicenda, talvolta in modo oppositivo, per giungere alla previsione finale. Anche nel caso dei modelli linguistici, tramite il *Supervised Dimension Projection Plot* e nuovamente gli SHAP Values si è cercato di analizzare più a fondo il funzionamento di questi strumenti, giungendo ad una rappresentazione abbastanza esaustiva delle relazioni tra le parole nelle due tipologie di testi.

Infine, nell'ultimo capitolo sono state messe alla prova le tecniche di classificazione, attraverso modifiche parziali ma mirate di alcune delle osservazioni presenti nel *test set*. Da questo processo è emersa una discreta robustezza dei modelli, che hanno risposto ai cambiamenti in modo abbastanza congruo alle aspettative. Questo lavoro ha mostrato come l'affinamento dei metodi e la creazione di variabili esplicative su misura permetta il corretto riconoscimento dei testi di classi diverse, lasciando spazio però al presentimento che con l'ininterrotto affinamento dei modelli linguistici e il progresso nel campo dell'intelligenza artificiale, diventerà sempre più difficile farlo.

Nello svolgimento delle analisi è emerso come il costo computazionale svolga un ruolo decisamente rilevante nello sviluppo di lavori di questo tipo: i tempi di elaborazione e le risorse necessarie sono stati il principale limite incontrato durante la stesura di questa tesi. A questo proposito, avendo a disposizione una maggiore capacità di calcolo sarebbe possibile perfezionare ulteriormente il processo di creazione degli *embedding*, valutando combinazioni diverse di strati nascosti e di metodi di aggregazione. Inoltre, adoperare griglie più fitte per i valori dei parametri durante l'addestramento dei modelli porterebbe a risultati più precisi. In aggiunta, si è notato come l'italiano ponga un consistente freno al numero di tecniche attualmente usufruibili per lo studio dei testi, dal momento che una buona parte di esse sono una prerogativa della lingua inglese. Miglioramenti che

non dipendono strettamente dalle risorse disponibili riguardano l'aggiunta di ulteriori indicatori linguistici e lessicali, volti a catturare aspetti dei testi non ancora analizzati; oppure, l'utilizzo di nuovi metodi di *Explainable Machine Learning* in grado di chiarire sotto altre prospettive il funzionamento delle tecniche meno interpretabili.

Infine, per ampliare le applicazioni pratiche dei risultati, sarebbe interessante mettere alla prova i modelli di classificazione tramite testi più brevi o significativamente alterati, per analizzare più approfonditamente le loro reazioni a tali variazioni. Un'altra idea potrebbe essere insegnare agli studenti come ottimizzare i contenuti generati da un'AI, per sfruttare al massimo un sistema che consente di risparmiare tempo e migliorare l'esecuzione di determinati compiti.

Bibliografia

Aas, K., Jullum, M., & Løland, A. (2021). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *Artificial Intelligence*, *298*, 103–502.

Allen, J. F. (2003). Natural language processing. In *Encyclopedia of computer science* (pp. 1218–1222).

Anderson, N., Belavy, D. L., Perle, S. M., Hendricks, S., Hespanhol, L., Verhagen, E., & Memon, A. R. (2023). AI did not write this manuscript, or did it? Can we trick the AI text detector into generated texts? The potential future of ChatGPT and AI in Sports & Exercise Medicine manuscript generation. *BMJ open sport & exercise medicine*, *9*(1), e001568.

Bengio, Y., Ducharme, R., & Vincent, P. (2003). A Neural Probabilistic Language Model. *Advances in Neural Information Processing Systems*, 932–938.

Binongo, J. (2003). Who Wrote the 15th Book of Oz? An Application of Multivariate Analysis to Authorship Attribution. *CHANCE*, *16*(2), 9–17. https://doi.org/10.1080/09332480.2003.10554843

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 427–431.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Boulton, R., & Betts, O. (2005). Snowball Stemming language and algorithms.

Bozkurt, I. N., Baghoglu, O., & Uyar, E. (2007). Authorship attribution. *2007 22nd International Symposium on Computer and Information Sciences*, 1–5.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger,

G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language Models are Few-Shot Learners.

Burrows, J. (2002). "Delta": a Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing*, *17*(3), 267–287. https://doi.org/10.1093/llc/17.3.267

Burrows, S., Uitdenbogerd, A. L., & Turpin, A. (2009). Application of information retrieval techniques for source code authorship attribution. *Database Systems for Advanced Applications: 14th International Conference, DASFAA 2009, Brisbane, Australia, April 21-23, 2009. Proceedings*, *14*, 699–713.

Chaka, C. (2023). Detecting AI content in responses generated by ChatGPT, YouChat, and Chatsonic: The case of five AI content detection tools. *Journal of Applied Learning and Teaching*, *6*(2).

Chan, C. (2023). grafzahl: fine-tuning Transformers for text data from within R. *Computational Communication Research*. https://doi.org/10.5117/CCR2023.1.003.CHAN

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://arxiv.org/abs/1603.02754

Chomsky, N. (1957). *Syntactic Structures*. Mouton.

Cortelazzo, M. A., & Tuzzi, A. (2018). *Drawing Elena Ferrante's profile: Workshop proceedings, Padova 7 September 2017*. Padova University Press.

Cui, A. (2023). *GPTzero Surpasses Competitors in Accuracies*. Recuperato marzo 18, 2024, da https://gptzero.me/news/gptzero-surpasses-competitors-in-accuracies

Dai, J., Pan, X., Sun, R., Ji, J., Xu, X., Liu, M., Wang, Y., & Yang, Y. (2023). Safe RLHF: Safe Reinforcement Learning from Human Feedback.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran & T. Solorio (Cur.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics.

Dunn, M., Terrill, A., Reesink, G., Foley, R., & Levinson, S. (2005). Structural Phylogenetics and the Reconstruction of Ancient Language History. *Science*, *309*(5743), 2072–2075. https://doi.org/10.1126/science.1114615

Eder, M. (2011). Style-markers in authorship attribution: a cross-language study of the authorial fingerprint. *Studies in Polish Linguistics*, *6*, 99–114. http://www.wuj.pl/page,art,artid,1923.html

Eder, M. (2013). Computational Stylistics and Biblical Translation: How Reliable Can a Dendrogram Be? In T. Piotrowski & Ł. Grabowski (Cur.), *The Translator and the Computer* (pp. 155–170). WSF Press.

Eder, M. (2017). Visualization in Stylometry: Cluster Analysis Using Networks. *Digital Scholarship in the Humanities*, *32*(1), 50–64.

Eder, M., Rybicki, J., & Kestemont, M. (2016a). Stylometry with R: A Package for Computational Text Analysis. *R J.*, *8*, 107. https://api.semanticscholar.org/CorpusID:18219905

Eder, M., Rybicki, J., & Kestemont, M. (2016b). Stylometry with R: a package for computational text analysis. *R Journal*, *8*(1), 107–121. https://journal.r-project.org/archive/2016/RJ-2016-007/index.html

Efron, B., & Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. CRC Press.

Erk, K. (2012). Vector Space Models of Word Meaning and Phrase Meaning: A Survey. *Language and Linguistics Compass*, *6*(10), 635–653. https://doi.org/10.1002/lnco.362

Ethayarajh, K. (2019). How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In K. Inui, J. Jiang, V. Ng & X. Wan (Cur.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 55–65). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1006

Facebook, I. (2016). *fastText: Library for fast text representation and classification*. https://github.com/facebookresearch/fastText

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In F. Palmer (Cur.), *Studies in linguistic analysis*. Philological Society (reprinted in 1968: Selected Papers of J. R. Firth).

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, *29*(5), 1189–1232.

Gaddis, G. M., & Gaddis, M. L. (1990). Introduction to biostatistics: Part 3, sensitivity, specificity, predictive value, and hypothesis testing. *Annals of emergency medicine*, *19*(5), 591–597.

Ganesan, A. V., Matero, M., Ravula, A. R., Vu, H., & Schwartz, H. A. (2021). Empirical Evaluation of Pre-trained Transformers for Human-Level NLP: The Role of Sample Size and Dimensionality. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4515–4532. https://doi.org/10.18653/v1/2021.naacl-main.357

Girosi, F., Jones, M., & Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural computation*, *7*(2), 219–269.

Goodwin, D. (2023). *OpenAI's AI Text Classifier no longer available due to 'low rate of accuracy'*. Recuperato luglio 26, 2023, da https://searchengineland.com/openai-ai-classifier-no-longer-available-429912

Greenacre, M. J. (1984). Theory and applications of correspondence analysis.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, *117*(4), 500–544.

Hothorn, T., Lausen, B., Benner, A., & Radespiel-Troeger, M. (2004). Bagging Survival Trees. *Statistics in Medicine*, *23*(1), 77–91.

Irvine, A., & Callison-Burch, C. (2017). A Comprehensive Analysis of Bilingual Lexicon Induction. *Computational Linguistics*, *43*(2), 273–310. https://doi.org/10.1162/COLI_a_00284

Juola, P., Sofko, J., & Brennan, P. (2006). A prototype for authorship attribution studies. *Literary and Linguistic Computing*, *21*(2), 169–178.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*, *103*, 102–274.

Khalil, M., & Er, E. (2023). Will ChatGPT get you caught? Rethinking of plagiarism detection. *arXiv preprint arXiv:2302.04335*.

Kjell, O., Giorgi, S., & Schwartz, H. (2023). The text-package: An R-package for analyzing and visualizing human language using natural language processing and transformers. *Psychological Methods*, *28*(6), 1478–1498.

Kneser, R., & Ney, H. (1995). Improved backing-off for M-gram language modeling. *1995 International Conference on Acoustics, Speech, and Signal Processing*, *1*, 181–184 vol.1. https://doi.org/10.1109/ICASSP.1995.479394

Kubrick, S. (1968). 2001: A Space Odyssey [Produced by Metro-Goldwyn-Mayer (MGM) and Stanley Kubrick Productions].

Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of statistical software*, *28*, 1–26.

Kuhn, M. (2014). Futility Analysis in the Cross-Validation of Machine Learning Models. https://arxiv.org/abs/1405.6974

Lambda Labs. (2024). *GPT-3: A Hitchhiker's Guide*. Recuperato giugno 22, 2024, da https://lambdalabs.com/blog/gpt-3

Lin, X., Wang, W., Li, Y., Yang, S., Feng, F., Wei, Y., & Chua, T.-S. (2024). Data-efficient Fine-tuning for LLM-based Recommendation.

Liu, Y., He, H., Han, T., Zhang, X., Liu, M., Tian, J., Zhang, Y., Wang, J., Gao, X., Zhong, T., Pan, Y., Xu, S., Wu, Z., Liu, Z., Zhang, X., Zhang, S., Hu, X., Zhang, T., Qiang, N., . . . Ge, B. (2024). Understanding LLMs: A Comprehensive Overview from Training to Inference.

Lucisano, P., & Piemontese, M. E. (1988). Gulpease: una formula per la predizione della leggibilità di testi in lingua italiana. *Scuola e Città*, *1988*, 110–124.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*.

Maffei, D. (1980). *La Donazione di Costantino Nei Giuristi Medievali*. Giuffrè.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Matero, M., Idnani, A., Son, Y., Giorgi, S., Vu, H., Zamani, M., Limbachiya, P., Guntuku, S. C., & Schwartz, H. A. (2019). Suicide Risk Assessment with Multi-level Dual-Context Language and BERT. In K. Niederhoffer, K. Hollingshead, P. Resnik, R. Resnik & K. Loveys (Cur.), *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology* (pp. 39–44). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-3005

Matlin, M. W., & Stang, D. J. (1978). *The Pollyanna principle: Selectivity in language, memory, and thought*. Schenkman Publishing Company.

McDonald, D. D. (2010). Natural language generation. *Handbook of natural language processing*, *2*, 121–144.

Meister, C., & Cotterell, R. (2021). Language Model Evaluation Beyond Perplexity. In C. Zong, F. Xia, W. Li & R. Navigli (Cur.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 5328–5339). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.414

Mesnil, G., Deng, L., Gao, J., He, X., & Shen, Y. (2014). Learning Semantic Representations Using Convolutional Neural Networks for Web Search. *Microsoft Research*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems*, 3111–3119.

Mikros, G., Koursaris, A., Bilianos, D., & Markopoulos, G. (2023). AI-Writing Detection Using an Ensemble of Transformers and Stylometric Features. *IberLEF@SEPLN*.

Mikros, G. K., & Perifanos, K. (2013). Authorship Attribution in Greek Tweets Using Author's Multilevel N-Gram Profiles. *AAAI Spring Symposium: Analyzing Microtext*, 17–23.

Mohammad, S., & Turney, P. (2010). Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 26–34. https://aclanthology.org/W10-0204

Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, *29*(3), 436–465.

Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (2ª ed.). https://christophm.github.io/interpretable-ml-book

Molnar, C. (2023). Interpreting machine learning models with SHAP. *Lulu. com*.

Mosteller, F., & Wallace, D. L. (1963). Inference in an Authorship Problem: A Comparative Study of Discrimination Methods Applied to the Authorship of the Disputed Federalist Papers. *Journal of the American Statistical Association*, *58*(302), 275–309. https://doi.org/10.1080/01621459.1963.10500849

Murtagh, F., & Legendre, P. (2014). Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? *Journal of Classification*, *31*(3), 274–295. https://doi.org/10.1007/s00357-014-9161-z

Pal, M. (2005). Random forest classifier for remote sensing classification. *International journal of remote sensing*, *26*(1), 217–222.

Paradis, E., Claude, J., & Strimmer, K. (2004). APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, *20*, 289–290. https://doi.org/10.1093/bioinformatics/btg412

Pegoraro, A., Kumari, K., Fereidooni, H., & Sadeghi, A.-R. (2023). To ChatGPT, or not to ChatGPT: That is the question! *arXiv preprint arXiv:2304.01487*.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. http://www.aclweb.org/anthology/D14-1162

Pierrehumbert, J. B. (2012). Burstiness of verbs and derived nouns. In *Shall We Play the Festschrift Game? Essays on the Occasion of Lauri Carlson's 60th Birthday* (pp. 99–115). Springer.

Plutchik, R. (1980). A general psychoevolutionary theory of emotion. In R. Plutchik & H. Kellerman (Cur.), *Emotion: Theory, research and experience, Theories of emotion* (pp. 3–33, Vol. 1). Academic Press.

R: A Language and Environment for Statistical Computing. (2024). R Foundation for Statistical Computing. Vienna, Austria. https://www.r-project.org/

Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245

Rajapakse, T. (2022). Simple Transformers.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Rider, P. R., & Yule, G. U. (1944). The statistical study of literary vocabulary. https://api.semanticscholar.org/CorpusID:86556814

Riloff, E. (1995). Little words can make a big difference for text classification. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 130–136. https://doi.org/10.1145/215206.215349

Rogers, R. (2022). *How to Spot Generative AI Text*. Recuperato marzo 18, 2024, da https://www.wired.com/story/how-to-spot-generative-ai-text-chatgpt/

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Schäfer, R. (2016). CommonCOW: Massively Huge Web Corpora from CommonCrawl Data and a Method to Distribute them Freely under Restrictive EU Copyright Laws. In N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Mae-

gaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk & S. Piperidis (Cur.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 4500–4504). European Language Resources Association (ELRA). https://aclanthology.org/L16-1712

Schmauder, A. R., Morris, R. K., & Poynor, D. V. (2000). Lexical processing and text integration of function and content words: Evidence from priming and eye fixations. *Memory & Cognition*, *28*(7), 1098–1108. https://doi.org/10.3758/BF03211811

Scholkopf, B., Sung, K.-K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T., & Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, *45*(11), 2758–2765.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms.

Seligman, M. (2015). The Arborist: a High-Performance Random Forest Implementation [Accessed: 2024-07-15].

Selj, V., Peng, F., Cercone, N., & Thomas, C. (2003). N-Gram-Based Author Profiles For Authorship Attribution. *Proceedings of the Conference Pacific Association for Computational Linguistics PACLING'03: 2003*.

Sellereite, N., Jullum, M., Olsen, L. H. B., Redelmeier, A., & Lachmann, J. (2024). *shapr: Prediction Explanation with Dependence-Aware Shapley Values* [R package version 0.2.3.9200, https://github.com/NorskRegnesentral/shapr/]. https://norskregnesentral.github.io/shapr/

Shields, C. (2006). *Mockingbird - A Portrait of Harper Lee*. Henry Holt.

Shijaku, R., & Canhasi, E. (2023). ChatGPT generated text detection. *Publisher: Unpublished*. https://doi.org/10.13140/RG.2.2.21317.52960

Shravan Kumar, B., & Ravi, V. (2017). Text Document Classification with PCA and One-Class SVM. In S. Satapathy, V. Bhateja, S. Udgata & P. Pattnaik (Cur.), *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications* (pp. 113–119, Vol. 515). Springer. https://doi.org/10.1007/978-981-10-3153-3_11

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu & S. Bethard (Cur.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1631–1642). Association for Computational Linguistics. https://aclanthology.org/D13-1170

Solnyshkina, M., et al. (2017). Evaluating Text Complexity and Flesch-Kincaid Grade Level. *Journal of Social Studies Education Research*, *8*(3), 238–248.

Spagnuolo, E. (2023). *Come riconoscere un testo scritto da ChatGPT* [Accessed: 2023-07-19]. https://www.wired.it/article/riconoscere-testo-scritto-chatgpt/

Stanikūnas, D., Mandravickaitė, J., & Krilavičius, T. (2017). Comparison of distance and similarity measures for stylometric analysis of Lithuanian texts. *CEUR Workshop proceedings*, 1–7.

Straka, M., & Straková, J. (2017). Universal Dependencies 2.0 Models for UDPipe (2017-08-01) [LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University]. http://hdl.handle.net/11234/1-2364

Sutton, C. D. (2005). Classification and regression trees, bagging, and boosting. *Handbook of statistics*, *24*, 303–329.

Tanaka-Ishii, K., & Aihara, S. (2015). Computational Constancy Measures of Texts—Yule's K and Rényi's Entropy. *Computational Linguistics*, *41*(3), 481–502. https://doi.org/10.1162/COLI_a_00228

Turing, A. M. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, *59*(236), 433–460.

Tuzzi, A. (2005). Analisi statistica del contenuto. In *Percorsi di ricerca sociale* (pp. 237–254). Carocci.

van Rossum, G. (1995). *Python reference manual* (R 9525).

Vapnik, V. N., Vapnik, V., et al. (1998). *Statistical learning theory*. wiley New York.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, J., Liu, S., Xie, X., & Li, Y. (2023). Evaluating AIGC Detectors on Code Content. *arXiv preprint arXiv:2304.05193*.

Ward, J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*, 236–244.

Weber-Wulff, D., Anohina-Naumeca, A., Bjelobaba, S., Foltýnek, T., Guerrero-Dib, J., Popoola, O., Šigut, P., & Waddington, L. (2023). Testing of detection tools for AI-generated text. *International Journal for Educational Integrity*, *19*(1), 26.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 1–3.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1-3), 37–52. https://doi.org/10.1016/0169-7439(87)80084-9

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., & Funtowicz, M. (2019). Huggingface's transformers: State-of-the-art Natural Language Processing.

Wright, M. N., & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, *77*(1), 1–17. https://doi.org/10.18637/jss.v077.i01

Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., Zhou, J., Chen, S., Gui, T., Zhang, Q., & Huang, X. (2023). A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models.

Yousri, R., & Safwat, S. (2023). How Big Can It Get? A comparative analysis of LLMs in architecture and scaling. *2023 International Conference on Computer and Applications (ICCA)*, 1–5. https://doi.org/10.1109/ICCA59364.2023.10401818

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zipf, G. K. (1949). *Human behavior and the principle of least effort*. Addison-Wesley Press.

Appendice

Modelli e Tecniche di Machine Learning

In questa sezione viene descritto nel dettaglio il funzionamento di alcune tecniche e modelli di classificazione utilizzati nel Capitolo 3.

A. Il Modello *Random Forest*

Il modello *Random Forest* (Brieman, 2001) è un classificatore composto da una collezione di classificatori ad albero $\{h(\mathbf{x}, \Theta_k), k = 1, \ldots\}$ dove i $\{\Theta_k\}$ sono vettori casuali indipendenti e identicamente distribuiti. Ogni albero contribuisce con un voto per la classe più popolare per l'*input* $\mathbf{x}$.

Costruzione degli Alberi

Per ogni albero k nella foresta, un vettore casuale Θ_k viene generato indipendentemente dai vettori casuali precedenti $\Theta_1, \ldots, \Theta_{k-1}$, ma con la stessa distribuzione. Un albero viene cresciuto usando il *set* di addestramento e Θ_k, risultando in un classificatore $h(\mathbf{x}, \Theta_k)$, dove $\mathbf{x}$ è un vettore di *input*. Dopo aver generato un gran numero di alberi, essi votano per la classe più popolare. Questi processi sono chiamati *foreste casuali*.

Accuratezza

Dato un insieme di classificatori $h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_K(\mathbf{x})$, e un *set* $\mathbf{X}$ estratto casualmente dalla distribuzione di un vettore casuale Y, la funzione margine è definita come:

$$mg(\mathbf{X}, Y) = av_k I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k I(h_k(\mathbf{X}) = j)$$

dove $I(\cdot)$ è la funzione indicatrice. La funzione margine misura l'estensione con cui il numero medio di voti in $\mathbf{X}$ per la classe corretta Y eccede il numero medio di voti per qualsiasi altra classe. L'errore di generalizzazione è dato da:

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X}, Y) < 0)$$

dove i pedici $\mathbf{X}, Y$ indicano che la probabilità è definita nello spazio $\mathbf{X}, Y$.
Nelle foreste casuali, $h_k(\mathbf{X}) = h(\mathbf{X}, \Theta_k)$. Per un grande numero di alberi, segue dalla Legge Forte dei Grandi Numeri e dalla struttura dell'albero che:

Teorema 1 *Man mano che il numero di alberi aumenta, per quasi tutti i* $\Theta_1, \ldots, \Theta_k$, PE^* *converge a:*

$$P_{\mathbf{X},Y}\left(P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j) < 0\right)$$

Questo risultato spiega perché le foreste casuali non si adattano eccessivamente man mano che vengono aggiunti più alberi, ma producono un valore limite dell'errore di generalizzazione.

Forza e Correlazione

Per le foreste casuali, si può derivare un limite superiore per l'errore di generalizzazione in termini di due parametri che misurano quanto sono accurate le singole classificazioni degli alberi e la dipendenza tra loro. Questi due parametri forniscono le basi per comprendere il funzionamento delle foreste casuali.
Definita la funzione margine per una *Random Forest* come:

$$mr(\mathbf{X}, Y) = P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j)$$

e la forza del *set* di classificatori $\{h(\mathbf{x}, \Theta)\}$ come:

$$s = E_{\mathbf{X},Y}[mr(\mathbf{X}, Y)]$$

e assumendo $s \geq 0$, la disuguaglianza di Chebychev fornisce:

$$PE^* \leq \frac{\text{var}(mr)}{s^2}.$$

Una espressione più rivelatrice per la varianza di mr è data dai passaggi seguenti. Sia

$$\hat{j}(\mathbf{X}, Y) = \arg\max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j)$$

che implica

$$\begin{aligned} mr(\mathbf{X}, Y) &= P_\Theta(h(\mathbf{X}, \Theta) = Y) - P_\Theta(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y)) \\ &= E_\Theta[I(h(\mathbf{X}, \Theta) = Y) - I(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y)]. \end{aligned}$$

La funzione margine grezza è data da:

$$rmg(\Theta, \mathbf{X}, Y) = I\left(h(\mathbf{X}, \Theta) = Y\right) - I\left(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y)\right).$$

Così, $mr(\mathbf{X}, Y)$ è il valore atteso di $rmg(\theta, \mathbf{X}, Y)$ rispetto a Θ. Per ogni funzione f l'identità

$$[E_\Theta f(\Theta)]^2 = E_{\Theta,\Theta'} f(\Theta) f(\Theta')$$

è valida quando Θ, Θ' sono indipendenti con la stessa distribuzione, che implica:

$$mr(\mathbf{X}, Y)^2 = E_{\Theta,\Theta'} rmg(\Theta, \mathbf{X}, Y) rmg(\Theta', \mathbf{X}, Y).$$

Utilizzando questa identità, si ottiene:

$$\begin{aligned} \text{var}(mr) &= E_{\Theta,\Theta'}(\text{cov}_{\mathbf{X},Y} rmg(\Theta, \mathbf{X}, Y) rmg(\Theta', \mathbf{X}, Y)) \\ &= E_{\Theta,\Theta'}(\rho(\Theta, \Theta') sd(\Theta) sd(\Theta')) \end{aligned}$$

dove $\rho(\Theta, \Theta')$ è la correlazione tra $rmg(\Theta, X, Y)$ e $rmg(\Theta', X, Y)$ con Θ, Θ' fissate e $sd(\theta$ deviazione standard di $rmg(\Theta', X, Y)$ tenendo Θ fissato. Dunque:

$$\begin{aligned} \text{var}(mr) &= \bar{\rho}(E_\Theta sd(\Theta))^2 \\ &\leq \bar{\rho} E_\Theta \text{var}(\Theta) \end{aligned}$$

dove $\bar{\rho}$ è il valore medio della correlazione, ovvero:

$$\bar{\rho} = E_{\Theta,\Theta'}(\rho(\Theta, \Theta') sd(\Theta) sd(\Theta')) / E_{\Theta,\Theta'}(sd(\Theta) sd(\Theta')).$$

Pertanto, si può scrivere:

$$\begin{aligned} E_{\Theta}\text{var}(\Theta) &\leq E_{\Theta}(E_{\text{X},Y}rmg(\Theta, \mathbf{X}, Y))^2 - s^2 \\ &\leq 1 - s^2. \end{aligned}$$

Questi passaggi portano a definire il limite superiore per l'errore di generalizzazione come

$$PE^* \leq \bar{\rho}(1 - s^2)/s^2.$$

Questa definizione mostra che i due ingredienti nell'errore di generalizzazione per le foreste casuali sono la forza dei classificatori singoli nella foresta e la correlazione tra loro in termini di funzioni di margine grezzo.

B. La tecnica del *Gradient Boosting*

Il *Gradient Boosting* (Friedman, 2001) è una tecnica di apprendimento d'insieme, in cui viene combinata una serie di modelli più semplici, detti *weak learners* o *base learners*, per costruire un modello più forte. Solitamente, questi *weak learners* sono alberi decisionali di piccole dimensioni.

La stima della funzione

Nel problema della stima della funzione (o "apprendimento predittivo"), si ha un sistema composto da una variabile casuale *output* o risposta y e un insieme di variabili casuali *input* o esplicative $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$. Usando un campione di addestramento $\{y_i, \mathbf{x}_i\}_{i=1}^N$ di valori y e $\mathbf{x}$ conosciuti, l'obiettivo è ottenere una stima o approssimazione $\widehat{F}(\mathbf{x})$, della funzione $F^*(\mathbf{x})$ che mappa $\mathbf{x}$ in y, minimizzando il valore atteso di una funzione di perdita specificata $L(y, F(\mathbf{x}))$ rispetto alla distribuzione congiunta di tutti i valori $(y, \mathbf{x})$:

$$F^* = \arg\min_F E_{y,\mathbf{x}}[L(y, F(\mathbf{x}))] = \arg\min_F E_{\mathbf{x}}\left[E_y(L(y, F(\mathbf{x}))) \,|\, \mathbf{x}\right].$$

Le funzioni di perdita $L(y, F)$ più comunemente utilizzate includono l'errore quadratico $(y - F)^2$ e l'errore assoluto $|y - F|$ per $y \in \mathbb{R}^1$ (regressione), e la log-verosimiglianza binomiale negativa $\log(1 + e^{-2yF})$ quando $y \in \{-1, 1\}$ (classificazione).
Il modello $F(x)$ è spesso ristretto a una classe parametrizzata di funzioni $F(\mathbf{x}; \mathbf{P})$, dove $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \dots\}$ è un insieme finito di parametri che identificano i singoli membri della classe. Questo approccio si focalizza su espansioni additive del tipo:

$$F(\mathbf{x}; \{\beta_m, \mathbf{a}_m\}_1^M) = \sum_{m=1}^{M} \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

dove $h(\mathbf{x}; \mathbf{a})$ è solitamente una funzione parametrizzata semplice delle variabili di input $\mathbf{x}$, caratterizzata dai parametri $\mathbf{a} = \{\alpha_1, \alpha_2, \dots\}$. Ogni termine differisce nei valori congiunti $\mathbf{a}_m$ scelti per questi parametri. Le espansioni additive di questo tipo sono alla base di molti metodi di approssimazione funzionale, come le reti neurali, le funzioni a base radiale, i modelli MARS e le *support vector machines*. Un caso di particolare interesse è quando ciascuna delle funzioni $h(\mathbf{x}; \mathbf{a}_m)$ è un piccolo albero di regressione.

Ottimizzazione Numerica

Scegliendo un modello parametrizzato $F(\mathbf{x};\mathbf{P})$, il problema di ottimizzazione funzionale viene ridotto a un problema di ottimizzazione dei parametri:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \phi(\mathbf{P}),$$

dove

$$\phi(\mathbf{P}) = E_{y,\mathbf{x}} L(y, F(\mathbf{x};\mathbf{P}))$$

e quindi

$$F^*(\mathbf{x}) = F(\mathbf{x};\mathbf{P}^*).$$

Per la maggior parte di $F(\mathbf{x},\mathbf{P})$ e L vengono usati metodi di ottimizzazione numerica per risolvere il problema di minimizzazione. Questo spesso consiste nell'esprimere la soluzione per i parametri nella forma

$$\mathbf{P}^* = \sum_{m=0}^{M} \mathbf{p}_m$$

dove $\mathbf{p}_0$ è un valore di supposizione iniziale e $\{\mathbf{p}_m\}_1^M$ sono i successivi incrementi (detti anche *step* o *boost*), ognuno basato sulla sequenza dei passi precedenti. Il modo per calcolare ciascun passo $\mathbf{p}_m$ è definito dal metodo di ottimizzazione *Steepest-descent*.

Steepest-descent è uno dei metodi di minimizzazione più semplici e più frequentemente usati. Esso definisce gli incrementi $\{\mathbf{p}_m\}_1^M$ come segue. Innanzitutto viene calcolato il gradiente $\mathbf{g}_m$ corrente:

$$\mathbf{g}_m = \{g_{jm}\} = \left\{ \left[\frac{\partial \Phi(\mathbf{P})}{\partial P_j} \right]_{\mathbf{P}=\mathbf{P}_{m-1}} \right\}$$

dove

$$\mathbf{P}_{m-1} = \sum_{i=0}^{m-1} \mathbf{p}_i.$$

Lo *step* è calcolato per essere

$$\mathbf{p}_m = -\rho_m \mathbf{g}_m$$

dove

$$\arg\min_{\rho} \Phi(\mathbf{P}_{m-1} - \rho \mathbf{g}_m).$$

Si dice che il gradiente negativo $-\mathbf{g}_m$ definisca la direzione della "discesa più ripida" e la

ricerca di ρ_m è chiamata "ricerca della linea" lungo quella direzione.

Ottimizzazione Numerica nello Spazio delle Funzioni

Invece di lavorare nello spazio dei parametri, si può adottare un approccio non parametrico e applicare l'ottimizzazione numerica direttamente nello spazio delle funzioni. Si considera $F(\mathbf{x})$ valutato in ogni punto $\mathbf{x}$ come un "parametro" e si cerca di minimizzare

$$\Phi(F) = E_{y,\mathbf{x}} L(y, F(\mathbf{x})) = E_{\mathbf{x}}[E_y(L(y, F(\mathbf{x}))) \mid \mathbf{x}]$$

o equivalentemente,

$$\phi(F(\mathbf{x})) = E_y[L(y, F(\mathbf{x})) \mid \mathbf{x}]$$

per ogni singola $\mathbf{x}$, direttamente rispetto a $F(\mathbf{x})$. Seguendo il paradigma dell'ottimizzazione numerica, si assume che la soluzione sia

$$F^*(\mathbf{x}) = \sum_{m=0}^{M} f_m(\mathbf{x})$$

dove $f_0(\mathbf{x})$ è un valore iniziale, e $\{f_m(\mathbf{x})\}_1^M$ sono funzioni incrementali (*step* o *boost*), definite dal metodo di ottimizzazione.

Per il metodo *steepest-descent*,

$$f_m(x) = -\rho_m g_m(\mathbf{x})$$

con

$$g_m(\mathbf{x}) = \left[\frac{\partial \phi(F(\mathbf{x}))}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = \left[\frac{\partial E_y[L(y, F(\mathbf{x})) \mid \mathbf{x}]}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

e

$$F_{m-1}(\mathbf{x}) = \sum_{i=0}^{m-1} f_i(\mathbf{x}).$$

Assumendo una regolarità sufficiente da poter scambiare differenziazione e integrazione, questo diventa

$$g_m(\mathbf{x}) = E_y\Big[\frac{\partial L(y, F(\mathbf{x}))}{\partial F(\mathbf{x})} \mid \mathbf{x}\Big]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}.$$

Il moltiplicatore ρ_m è dato dalla ricerca della linea

$$\arg\min_{\rho} E_{y,\mathbf{x}} L(y, F_{m-1}(\mathbf{x}) - \rho g_m(\mathbf{x})).$$

Implementazione per dati finiti

In contesti reali, l'algoritmo di *Gradient Boosting* non opera su una distribuzione congiunta completa $(y, \mathbf{x})$, ma piuttosto su un campione finito di dati $\{(y_i, \mathbf{x}_i)\}_1^N$. In questo caso $E_y[\cdot \mid \mathbf{x}]$ non può essere stimato accuratamente dai suoi valori per ogni $\mathbf{x}$, e anche se fosse possibile, si vorrebbe stimare $F^*(\mathbf{x})$ in valori $\mathbf{x}$ diversi dai quelli dei punti di addestramento. Il supporto deve essere preso in prestito dai punti dati vicini imponendo la liscezza della soluzione. Un modo per fare ciò è assumere una forma parametrizzata ed eseguire l'ottimizzazione dei parametri per minimizzare la stima basata sui dati della perdita attesa corrispondente,

$$\{\beta_m, \mathbf{a}_m\}_1^M = \underset{\{\beta'_m, \mathbf{a}'_m\}_1^M}{\arg\min} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta'_m h(\mathbf{x}_i; \mathbf{a}'_m)\right).$$

In situazioni dove ciò non è realizzabile si può tentare un approccio *greedy stagewise*. Per $m = 1, 2, \ldots, M$,

$$(\beta_m, \mathbf{a}_m) = \underset{\beta, \mathbf{a}}{\arg\min} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \beta h(\mathbf{x}_i; \mathbf{a}))$$

e in seguito

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x}; \mathbf{a}_m).$$

Questa stategia *stagewise* è diversa dall'approccio *stepwise* che ricalibra i termini precedentemente inseriti quando vengono aggiunti nuovi termini.

In ambito di *machine learning* l'approccio *greedy stagewise* è denominato *boosting* dove $y \in \{-1, 1\}$ e $L(y, F)$ è un criterio di perdita esponenziale e^{-yF} oppure una verosimiglianza binomiale negativa. La funzione $h(\mathbf{x}; \mathbf{a})$ è chiamata *weak learner* o *base learner* ed è solitamente un albero di classificazione.

Si supponga che per una particolare funzione di perdita $L(y, F)$ e/o *base learner* $h(\mathbf{x}; \mathbf{a})$ la soluzione al problema di minimizzazione sia difficile da ottenere. Dato un qualsiasi approssimatore $F_{m-1}(\mathbf{x})$, la funzione $\beta_m h(\mathbf{x}; \mathbf{a}_m)$ può essere vista come il miglior passo *greedy* verso la stima basata sui dati di $F^*(\mathbf{x})$, sotto il vincolo che la "direzione" $h(\mathbf{x}; \mathbf{a})$ del passo sia un membro della classe parametrizzata di funzioni $h(\mathbf{x}; \mathbf{a})$. Può quindi essere considerato come un passo della *steepest descent* sotto tale vincolo.

Per costruzione, l'analogo basato sui dati del gradiente negativo non vincolato,

$$-g_m(\mathbf{x}_i) = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

fornisce la migliore direzione di discesa ripida $-\mathbf{g}_m = \{-g_m(\mathbf{x}_i)\}_1^N$ nello spazio dei dati N-dimensionale a $F_{m-1}(\mathbf{x})$. Tuttavia, questo gradiente è definito solo nei punti $\{\mathbf{x}_i\}_1^N$ e non può essere generalizzato ad altri valori di $\mathbf{x}$. Una possibilità per la generalizzazione è scegliere quel membro della classe parametrizzata $h(\mathbf{x}; \mathbf{a}_m)$ che produce $\mathbf{h}_m = \{h(\mathbf{x}_i; \mathbf{a}_m)\}_1^N$ il più parallelo possibile a $-\mathbf{g}_m \in \mathbb{R}^N$. Questo è l'elemento $h(\mathbf{x}; \mathbf{a})$ maggiormente correlato con $-g_m(\mathbf{x})$ rispetto alla distribuzione dei dati. Può essere ottenuto dalla soluzione

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_1^N \left[-g_m(\mathbf{x}_i) - \beta h(\mathbf{x}_i; \mathbf{a})\right]^2 .$$

Questo gradiente negativo vincolato $h(\mathbf{x}; \mathbf{a}_m)$ viene utilizzato al posto del gradiente negativo non vincolato $-\mathbf{g}_m(\mathbf{x})$ nella strategia *steepest-descent*. Specificamente, la ricerca della linea è effettuata tramite

$$\rho_m = \arg\min_{\rho} \sum_1^N L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$$

e l'approssimazione aggiornata mediante

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m).$$

Fondamentalmente, invece di ottenere la soluzione sotto un vincolo di continuità liscio, il vincolo è applicato alla soluzione non vincolata (ruvida) adattando $h(\mathbf{x}; \mathbf{a})$ ai "pseudo-risultati" $\{\tilde{y}_i = -g_m(\mathbf{x}_i)\}_{i=1}^N$. Questo permette di sostituire il difficile problema di minimizzazione con la minimizzazione di una funzione tramite minimi quadrati, seguita poi dall'ottimizzazione di un singolo parametro basata sul criterio originale.

Algoritmo 3 Gradient Boosting

1: $F_0(\mathbf{x}) = \arg\min_{\rho} \sum_{i=1}^N L(y_i, \rho)$
2: **for** $m = 1$ to M **do**
3: **for** $i = 1$ to N **do**
4: $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$
5: **end for**
6: $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^N \left[\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})\right]^2$
7: $\rho_m = \arg\min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
8: $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
9: **end for**

Pertanto, per qualsiasi $h(\mathbf{x}; \mathbf{a})$ per il quale esiste un algoritmo di minimi quadrati fattibile per risolvere il problema, si può usare questo approccio per minimizzare qualsiasi funzione di perdita $L(y, F)$ differenziabile, in combinazione con la modellazione additiva *forward stagewise*. Questo porta alla definizione (generica) dell'algoritmo 3 usando la tecnica *steepest-descent*.

In realtà qualsiasi criterio di *fitting* che stima il valore atteso condizionato (dato $\mathbf{x}$) potrebbe, in linea di principio, essere utilizzato per stimare il gradiente negativo *smoothed* alla riga 6 dell'algoritmo. I minimi quadrati sono una scelta naturale per via delle loro buone proprietà computazionali.

C. *Support Vector Machines* con *kernel* radiale

Le *Support Vector Machines* (SVM) sono modelli di apprendimento supervisionato con l'obiettivo principale di trovare il miglior iperpiano che separa i dati in categorie diverse. Le SVM cercano di massimizzare il margine, ovvero la distanza tra il confine di separazione (iperpiano) e i punti d'esempio (*input*) più vicini di ciascuna categoria, chiamati vettori di supporto. Le SVM possono essere estese per gestire problemi non lineari utilizzando tecniche come il *kernel trick*, che proietta i dati in uno spazio ad alta dimensione dove un iperpiano lineare può separarli facilmente (Scholkopf *et al.*, 1997).

Minimizzazione del rischio strutturale

Nel caso del riconoscimento di *pattern* a due classi, il compito di apprendere dagli esempi può essere formulato nel modo seguente. Dato un *set* di funzioni

$$\{f_\alpha \colon \alpha \in \Lambda\}, \quad f_\alpha \colon \mathbf{R}^N \to \{-1, +1\}$$

e un *set* di esempi

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell), \quad \mathbf{x}_i \in \mathbf{R}^N, \quad y_i \in \{-1, +1\}$$

dove ognuno di essi è generato da una distribuzione di probabilità ignota $P(\mathbf{x}, y)$, si vuole trovare una funzione f_{α^*} che fornisca il valore più piccolo possibile per l'errore medio commesso su nuovi esempi estratti casualmente da P chiamato rischio

$$R(\alpha) = \int \frac{1}{2} \left| f_\alpha(\mathbf{x}) - y \right| dP(\mathbf{x}, y).$$

Il problema è che $R(\alpha)$ è ignoto dal momento che lo è anche $P(\mathbf{x}, y)$. Di conseguenza è necessario utilizzare il principio di induzione per minimizzare il rischio.

L'approccio diretto per minimizzare il *rischio empirico*

$$R_{\text{emp}}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{1}{2} \left| f_\alpha(\mathbf{x}_i) - y_i \right|$$

risulta non garantire un piccolo rischio effettivo (cioè, un piccolo errore sul *training set* non implica un piccolo errore sul *test set*) se il numero l di esempi di *training* è limitato.

Tra i metodi esistenti per sfruttare al meglio una quantità limitata di dati, il principio di *minimizzazione del rischio strutturale* si basa sul fatto che per questo problema di apprendimento, per ogni $\alpha \in \Lambda$ con una probabilità di almeno $1 - \eta$, il limite

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \phi\left[\frac{h}{\ell}\right]$$

è valido, dove ϕ è definito come

$$\phi\left[\frac{h}{\ell}\right] = \sqrt{\frac{h\left(\log\frac{2\ell}{h} + 1\right) - \log(\eta/4)}{\ell}}.$$

Il parametro h descrive la capacità di un insieme di funzioni implementabili dalla macchina di apprendimento. Per la classificazione binaria, rappresenta il numero massimo di punti che possono essere separati in due classi in tutti i 2^h modi possibili utilizzando le funzioni della macchina di apprendimento; cioè, per ciascuna separazione possibile, esiste una funzione che assume il valore 1 su una classe e -1 sull'altra classe.

Secondo il limite definito, dato un numero ℓ di esempi di *training*, si può controllare il rischio tramite due quantità: $R_emp(\alpha)$ e $h(\{f_\alpha \colon \alpha \in \Lambda'\})$, con Λ' che indica alcuni sottoinsiemi del *set* Λ di indici. Il rischio empirico dipende dalla funzione scelta dalla macchina di apprendimento (cioè da α), e può essere controllato scegliendo il corretto α. Invece, h dipende dall'insieme di funzioni $\{f_\alpha \colon \alpha \in \Lambda'\}$ che la macchina di apprendimento può implementare. Per controllare h, si introduce una struttura di sottoinsiemi nidificati $S_n \colon = \{f_\alpha \colon \alpha \in \Lambda_n\}$ di $\{f_\alpha \colon \alpha \in \Lambda\}$

$$S_1 \subset S_2 \subset \cdots \subset S_n \subset \cdots$$

le cui dimensioni h, di conseguenza, soddisfano

$$h_1 \leq h_2 \leq \cdots \leq h_n \leq \cdots$$

Per un dato insieme di osservazioni $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ il principio di minimizzazione del rischio strutturale sceglie la funzione $f_{\alpha_l^n}$ nel sottoinsieme $\{f_\alpha \colon \alpha \in \Lambda_n\}$ per cui il limite garantito del rischio è minimo.

Una struttura nel *set* di iperpiani

Ogni particolare scelta per la struttura di sottoinsiemi nidificati dà origine a un algoritmo di apprendimento. L'algoritmo delle SVM si basa su una struttura sull'insieme degli iperpiani separatori. Per descriverlo, si noti innanzitutto che dato uno spazio prodotto scalare $\mathbf{Z}$ e un insieme di vettori $\mathbf{z_1}, \ldots, \mathbf{z_r} \in \mathbf{Z}$, ogni iperpiano $\{\mathbf{z} \in \mathbf{Z} \colon (\mathbf{w} \cdot \mathbf{z}) + b = 0\}$ corrisponde a una coppia canonica $(\mathbf{w}, b) \in \mathbf{Z} \times \mathbf{R}$ se si richiede inoltre che

$$\min_{i=1,\cdots,r} |(\mathbf{w} \cdot \mathbf{z}_i) + b| = 1$$

cioè che la scala di $\mathbf{w}$ e b sia tale che il punto più vicino all'iperpiano abbia una distanza di $1/||\mathbf{w}||$. Sia R il raggio della più piccola palla $B_R(\mathbf{a}) = \{\mathbf{z} \in \mathbf{Z} : \|\mathbf{z} - \mathbf{a}\| < R\}$ $(\mathbf{a} \in \mathbf{Z})$ contenente i punti $\mathbf{z}_1, \ldots, \mathbf{z}_r$, e

$$f_{\mathbf{w},b} = \mathrm{sgn}[(\mathbf{w} \cdot \mathbf{z}) + b]$$

dove la funzione decisionale è definita in questi punti. La possibilità di introdurre una struttura sull'insieme degli iperpiani è basata sulla loro proprietà secondo la quale l'insieme degli iperpiani canonici $\{f_{\mathbf{w},b} \colon \|\mathbf{w}\| \leq A\}$ ha una dimensione h che soddisfa

$$h < R^2 A^2 + 1.$$

L'algoritmo a vettori di supporto

Si supponga ora di voler trovare una funzione decisionale $f_{\mathbf{w},b}$ con la proprietà $f_{\mathbf{w},b}(\mathbf{z}_i) = y_i, i = \ldots, \ell$. Se questa funzione esiste, la canonicità $\min_{i=1,\ldots,r} |(\mathbf{w} \cdot \mathbf{z}_i) + b| = 1$ implica

$$y_i[(\mathbf{w} \cdot \mathbf{z}_i) + b] \geq 1, \quad i = 1, \ldots, \ell.$$

In pratica, spesso non esiste un iperpiano separatore. Per consentire la possibilità che alcuni esempi violino quest'ultima disuguaglianza, vengono introdotte variabili

$$\xi_i \geq 0, \quad i = 1, \ldots, \ell$$

per ottenere

$$y_i[(\mathbf{w} \cdot \mathbf{z}_i) + b] \geq 1 - \xi_i, \quad i = 1, \ldots, \ell.$$

L'approccio SVM per minimizzare il limite di rischio garantito $R(\alpha) \leq R_{\text{ernp}}(\alpha) + \phi\left[\frac{h}{\ell}\right]$ consiste nel seguente procedimento. Si minimizza

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + \gamma \sum_{i=1}^{\ell} \xi_i$$

soggetto ai vincoli precedentemente definiti. A causa di $h < R^2 A^2 + 1$, la minimizzazione del primo termine è collegata alla minimizzazione della dimensione h della classe di macchine di apprendimento considerate, che minimizza così anche il secondo termine del limite. Il termine $\sum_{i=1}^{\ell} \xi_i$, d'altra parte, rappresenta un limite superiore sul numero di misclassificazioni nel *set* di addestramento, che controlla così il termine del rischio. Per una costante positiva adatta γ, questo approccio costituisce quindi un'implementazione pratica della minimizzazione del rischio strutturale sull'insieme di funzioni dato.

Introducendo i moltiplicatori di Lagrange α_i, è possibile dimostrare che la soluzione ha un'espansione

$$\mathbf{w} = \sum_{i=1}^{\ell} y_i \alpha_i \mathbf{z}_i$$

con coefficienti $\alpha_i \neq 0$ solo dove i corrispondenti dati di esempio $(\mathbf{z}_i, y_i)$ incontrano precisamente il vincolo $y_i[(\mathbf{w} \cdot \mathbf{z}_i) + b] \geq 1 - \xi_i, \quad i = 1, \ldots, \ell$. Questi $\mathbf{z}_i$ sono chiamati *vettori di supporto*. Tutti i restanti dati dell'insieme di addestramento sono irrilevanti. Il loro vincolo è soddisfatto automaticamente (con $\xi_i = 0$), ed essi non appaiono nell'espansione. I coefficienti α_i vengono trovati risolvendo il seguente problema di programmazione quadratica. Si massimizza

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (\mathbf{z}_i \cdot \mathbf{z}_j)$$

soggetto ai vincoli

$$0 \leq \alpha_i \leq \gamma, \quad i = 1, \ldots, \ell, \quad \text{e} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0.$$

Per linearità del prodotto scalare, la funzione decisionale $f_{\mathbf{w},b}$ può quindi essere scritta come

$$f(\mathbf{z}) = \text{sgn}\left[\sum_{i=1}^{\ell} y_i \alpha_i \cdot (\mathbf{z} \cdot \mathbf{z}_i) + b\right]$$

Per poter avere superfici decisionali più generali di quelle lineari, si può innanzitutto trasformare in modo non lineare un insieme di vettori di *input* $\mathbf{x}_1, \ldots, \mathbf{x}_l$ in uno spazio delle caratteristiche ad alta dimensione tramite una mappa $\Phi\colon \mathbf{x}_i \mapsto \mathbf{z}_i$ e poi fare lì una separazione lineare. Massimizzare $W(\alpha)$ e valutare $f(\mathbf{z})$ richiede quindi il calcolo dei prodotti scalari $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i))$ in uno spazio ad alta dimensione. Si è interessati ai casi in cui questi calcoli costosi possono essere ridotti significativamente usando una funzione adatta $\mathbf{K}$ tale che

$$(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)) = K(\mathbf{x}, \mathbf{x}_i)$$

porti a funzioni decisionali della forma

$$f(\mathbf{x}) = \operatorname{sgn}\left[\sum_{i=1}^{\ell} y_i \alpha_i \cdot K(\mathbf{x}, \mathbf{x}_i) + b\right].$$

In pratica, non è necessario preoccuparsi di concepire la mappa Φ. Si sceglierà un $\mathbf{K}$ che sia il *kernel* continuo di un operatore integrale positivo definito, che può essere espanso in una serie uniformemente convergente in termini di autofunzioni ψ_j e autovalori positivi λ_j

$$K(\mathbf{x}, \mathbf{x_i}) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{x_i})$$

corrispondente a un prodotto scalare $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x_i}))$ nel dominio (possibilmente infinito-dimensionale) di

$$\Phi : \mathbf{x} \mapsto \left[\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \ldots\right].$$

Di conseguenza, tutto ciò che è stato detto sul caso lineare si può applicare anche ai casi non lineari ottenuti utilizzando un opportuno *kernel* $\mathbf{K}$ invece del prodotto scalare euclideo. Si è ora in grado di spiegare come l'algoritmo SV possa costruire classificatori con funzione a base radiale, cioè utilizzando

$$K(\mathbf{x}, \mathbf{x_i}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x_i}\|^2}{c}\right).$$

Inoltre, per trovare la funzione decisionale $f(\mathbf{x})$ si massimizza

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

soggetto ai vincoli

$$0 \leq \alpha_i \leq \gamma, \quad i = 1, \ldots, \ell, \quad \text{e} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0.$$

Per ottenere la soglia b, per i vettori di supporto $\mathbf{x}_j$ per i quali $\xi_j = 0$, si adopera

$$\sum_{i=1}^{\ell} y_i \alpha_i \cdot K(\mathbf{x}_j, \mathbf{x}_i) + b = y_j.$$

D. Algoritmi per Lasso, Ridge ed *elastic net*

La classe `glmnet` comprende modelli lineari generalizzati (quindi anche di classificazione binaria) stimati tramite la massima verosimiglianza penalizzata. Il percorso di regolarizzazione viene calcolato per la penalizzazione *elastic net* su una griglia di valori (su scala logaritmica) per il parametro di regolarizzazione λ. Gli algoritmi di `glmnet` utilizzano ciclicamente l'algoritmo *coordinate descent*, che ottimizza successivamente la funzione obiettivo rispetto a ciascun parametro, mantenendo fissi gli altri, e ripete il ciclo fino alla convergenza (Friedman *et al.*, 2010).

Il problema di minimizzazione

Nel classico problema di regressione lineare si ha una variabile risposta $Y \in \mathbb{R}$ e un vettore predittore $X \in \mathbb{R}^p$, e si approssima la funzione di regressione con un modello lineare $E(Y \mid X = x) = \beta_0 + x^\top \beta$. Si hanno N coppie di osservazioni (x_i, y_i) e si supponga che i x_{ij} siano standardizzati: $\sum_{i=1}^N x_{ij} = 0$, $\frac{1}{N}\sum_{i=1}^N x_{ij}^2 = 1$, per $j = 1, \ldots, p$. L'*elastic net* risolve il seguente problema:

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} R_\lambda(\beta_0, \beta) = \min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{2N} \sum_{i=1}^N (y_i - \beta_0 - x_i^\top \beta)^2 + \lambda P_\alpha(\beta) \right],$$

dove

$$\begin{aligned} P_\alpha(\beta) = &\quad (1-\alpha)\frac{1}{2}||\beta||_{\ell_2}^2 + \alpha||\beta||_{\ell_1} \\ = &\quad \sum_{j=1}^p \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j| \right]. \end{aligned}$$

P_α è la penalità *elastic-net*, e rappresenta un compromesso tra la penalità di regressione Ridge ($\alpha = 0$) e la penalità Lasso ($\alpha = 1$). Questa penalità è particolarmente utile nella situazione $p \gg N$, o in qualsiasi situazione in cui ci siano molte variabili predittive correlate.

Per il risolvere il problema di minimizzazione è possibile utilizzare l'algoritmo *coordinate descent*. Si supponga di avere delle stime $\tilde{\beta}_0$ e $\tilde{\beta}_\ell$, e di voler ottimizzare parzialmente rispetto a β_j. Si vuole calcolare il gradiente in $\beta_j = \tilde{\beta}_j$, che esiste solo se $\tilde{\beta}_j \neq 0$. Se $\tilde{\beta}_j > 0$, allora

$$\frac{\partial R_\lambda}{\partial \beta_j}|_{\beta=\tilde{\beta}} = -\frac{1}{N}\sum_{i=1}^N x_{ij}(y_i - \tilde{\beta}_o - x_i^\top \tilde{\beta}) + \lambda(1-\alpha)\beta_j + \lambda\alpha.$$

L'espressione per $\tilde{\beta}_j < 0$ è simile per il caso $\tilde{\beta}_j = 0$ è trattato separatamente. L'aggiornamento *coordinate-wise* ha la forma

$$\tilde{\beta}_j \leftarrow \frac{S\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\alpha\right)}{1 + \lambda(1-\alpha)}$$

dove $\tilde{y}_i^{(j)} = \tilde{\beta}_0 + \sum_{(i)} x_{i\ell}\tilde{\beta}_\ell$ è il valore stimato escludendo il contributo di x_{ij}, e quindi $y_i - \tilde{y}_i^{(j)}$ è il residuo parziale della stima con β_j. A causa della standardizzazione, $\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \tilde{y}_i^{(j)})$ è il coefficiente dei minimi quadrati semplici quando si adatta questo residuo parziale a x_{ij}. Invece, $S(z, \gamma)$ è l'operatore *soft-tresholding* con valore

$$\text{sgn}(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma & \text{se } z > 0 \text{ e } \gamma < |z| \\ z + \gamma & \text{se } z < 0 \text{ e } \gamma < |z| \\ 0 & \text{se } \gamma \geq |z|. \end{cases}$$

Pertanto, si calcola il coefficiente dei minimi quadrati semplici sul residuo parziale, si applica la soglia per gestire il contributo della penalizzazione Lasso, e poi si applica una riduzione proporzionale per la penalizzazione Ridge.

Regressione logistica con regolarizzazione

Quando la variabile di risposta è binaria, si utilizza spesso il modello di regressione logistica lineare. Sia G la variabile di risposta, che assume valori in $\mathcal{G} = \{1, 2\}$. Il modello di regressione logistica rappresenta le probabilità condizionali della classe attraverso una funzione lineare dei predittori:

$$\Pr(G = 1|x) = \frac{1}{1 + e^{-(\beta_0 + x^\top \beta)}},$$

$$\Pr(G = 2|x) = \frac{e^{\beta_0 + x^\top \beta}}{1 + e^{\beta_0 + x^\top \beta}} = 1 - \Pr(G = 1|x).$$

In alternativa, questo implica che

$$\log\left(\frac{\Pr(G = 1|x)}{\Pr(G = 2|x)}\right) = \beta_0 + x^\top \beta.$$

Si può adattare questo modello tramite la massimizzazione regolarizzata della verosimiglianza (binomiale). Sia $p(x_i) = \Pr(G = 1|x_i)$ la probabilità per l'osservazione i a un particolare valore per i parametri (β_0, β), quindi si massimizza la verosimiglianza

penalizzata logaritmica

$$\max_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{N} \sum_{i=1}^{N} \{ I(g_i = 1) \log p(x_i) + I(g_i = 2) \log(1 - p(x_i)) \} - \lambda P_\alpha(\beta) \right].$$

Indicando $y_i = I(g_i = 1)$, la parte di verosimiglianza logaritmica dell'espressione precedente può essere scritta nella forma più esplicita

$$\ell(\beta_0, \beta) = \frac{1}{N} \sum_{i=1}^{N} y_i \cdot (\beta_0 + x_i^\top \beta) - \log(1 + e^{\beta_0 + x_i^\top \beta}),$$

una funzione concava dei parametri. L'algoritmo di Newton per massimizzare la verosimiglianza logaritmica (non penalizzata) consiste nel risolvere iterativamente sistemi di equazioni dei minimi quadrati pesati. Pertanto, se le stime correnti dei parametri sono $(\tilde{\beta}_0, \tilde{\beta})$, si crea un'approssimazione quadratica alla verosimiglianza logaritmica, pari a

$$\ell_Q(\beta_0, \beta) = -\frac{1}{2N} \sum_{i=1}^{N} w_i (z_i - \beta_0 - x_i^\top \beta)^2 + C(\tilde{\beta}_0, \tilde{\beta})^2$$

dove

$$\begin{aligned} z_i &= \tilde{\beta}_0 + x_i^\top \tilde{\beta} + \frac{y_i - \tilde{p}(x_i)}{\tilde{p}(x_i)(1 - \tilde{p}(x_i))}, \quad \text{(risposta corretta)} \\ w_i &= \tilde{p}(x_i)(1 - \tilde{p}(x_i)), \quad \text{(pesi)} \end{aligned}$$

e $\tilde{p}(x_i)$ è valutato ai parametri correnti. L'ultimo termine è costante. L'aggiornamento di Newton si ottiene minimizzando ℓ_Q.

L'approccio tramite *coordinate descent* è simile. Per ogni valore di λ, si crea un ciclo esterno che calcola l'approssimazione quadratica ℓ_Q sui parametri correnti $(\tilde{\beta}_0, \tilde{\beta})$. Si utilizza quindi la discesa a coordinate per risolvere il problema dei minimi quadrati pesati penalizzati

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \{ -\ell_Q(\beta_0, \beta) + \lambda P_\alpha(\beta) \}.$$

Questo comporta una sequenza di cicli annidati:

- **Ciclo esterno**: decremento di λ
- **Ciclo intermedio**: aggiornamento dell'approssimazione quadratica ℓ_Q utilizzando i parametri correnti $(\tilde{\beta}_0, \tilde{\beta})$
- **Ciclo interno**: esecuzione dell'algoritmo di *coordinate descent* sul problema dei minimi quadrati pesati penalizzati

che definiscono il processo.