UNIVERSITÀ DI PISA

DIPARTIMENTO DI
FILOLOGIA, LETTERATURA E LINGUISTICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA UMANISTICA

TESI DI LAUREA

Accettabilità e complessità linguistica all'interno di un Neural Language Model

Candidato:
Lucia Pifferi

Relatore:
Prof. Felice Dell'Orletta

Controrelatore:
Prof. Alessandro Lenci

Anno Accademico 2020/2021

A mio cugino Alessandro,
ricordandolo sempre con affetto

Indice

1. Introduzione 5

2. Stato dell'arte sui modelli di rappresentazione del linguaggio 8

2.1 Prospettiva storica 8
2.1.1 I modelli count 9
2.1.2 I modelli predict 10
2.1.3 Limiti di queste prime due classi di modelli 12
2.1.4 I modelli distribuzionali contestualizzati 16
2.1.5 Transformer 16
2.1.6 BERT 24
2.1.7 Limiti di BERT 28

2.2 Profilazione linguistica di un modello neurale del linguaggio 30

3. Il compito dell'accettabilità/complessità linguistica 38

3.1 Task AcCompl-it 39
3.1.1 Dataset AcCompl-it 40
3.1.2 Risultati AcCompl-it 45

3.2 Dataset selezionati per questo studio 47
3.2.1 Dati per la valutazione di accettabilità/complessità 47
3.2.2 Italian Dependency Treebank (IUDT) 51

4. Predizione dell'accettabilità e della complessità 53

4.1 Analisi con un regressore lineare 53

4.2 Analisi con un modello neurale del linguaggio: BERT 58
4.2.1 Transfer learning 61

5. Studio sulle rappresentazioni di BERT 64

5.1 Similarità tra le rappresentazioni dei modelli neurali 64

5.2 Profilazione linguistica dei modelli neurali 68

6. Conclusioni 83

7. Bibliografia 89

8. Sitografia 95

9. Appendice 96

9.1 Feature linguistiche estratte da Profiling UD 96

10. Ringraziamenti 101

«Language is an imperfect, incomplete, and low-bandwidth serialization protocol for the internal data structures we call thoughts.»

(Yann LeCun)

1. Introduzione

Il presente lavoro di tesi descrive uno studio sulla valutazione dell'accettabilità e la complessità linguistica di una frase utilizzando un *Neural Language Model* (BERT) e andando ad analizzare come le rappresentazioni interne alla rete neurale cambiano quando addestrata a predire la percezione di un parlante rispetto ai due fenomeni. Questa ricerca è stata svolta a seguito di un'esperienza di tirocinio presso l'Istituto di Linguistica Computazionale "Antonio Zampolli" del CNR di Pisa.

La scelta di questo argomento è stata motivata dalla necessità, sempre maggiore da parte di tutta la comunità scientifica, di risorse in grado di predire le due valutazioni a livello di frase. Infatti, queste si sono mostrate fondamentali per stimare la qualità di sistemi di generazione automatica (come sistemi di traduzione o semplificazione del testo) basati su algoritmi di *deep neural network* o per elaborare sistemi di valutazione di leggibilità utili a comprendere il migliore pubblico per un determinato testo.
Un'ulteriore motivazione che ci ha spinto a condurre questa ricerca è stata quella di avanzare la comprensione di come un modello neurale del linguaggio riesca a trarre le sue conclusioni e come elabori internamente l'informazione. Questo è un aspetto cruciale nel settore dell'intelligenza artificiale e sono oggi molteplici gli studi condotti nell'emergente campo dell'*explainable AI*, in cui questa ricerca si colloca. Infatti, comprendere a pieno le motivazioni dietro al responso ottenuto da una rete neurale può aiutarci a sviluppare modelli che siano sempre più efficaci e le cui decisioni siano più facilmente comprensibili.

Per il seguente studio, sono stati utilizzati sia un corpus valutato con giudizi di accettabilità e complessità linguistica rilasciato per il task *AcCompl-it* della campagna di valutazione EVALITA 2020, che sezioni dell'Italian Dependency Treebank (IUDT). In particolare, il primo è stato necessario per addestrare diversi modelli (che sfruttano o meno tecniche di *transfer learning*) sui compiti di valutazione dei due giudizi linguistici, il secondo, invece, per eseguire 82 *probing task* al fine di analizzare la conoscenza acquisita in ciascun layer e comprendere quanto differiscono le informazioni linguistiche immagazzinate prima e dopo il processo di fine tuning.

Inoltre, il corpus IUDT è stato utilizzato per estrarre diverse rappresentazioni delle frasi da ciascuno strato di ogni modello elaborato, per poi valutarne la similarità cosenica e comprendere quanto queste differiscano nelle reti messe direttamente a punto sui diversi compiti linguistici e in quelle che sfruttano metodi di *transfer learning*.

Nei successivi capitoli, prima verrà descritto lo stato dell'arte dei modelli di rappresentazione del linguaggio (nel capitolo 2), poi verranno affrontate le domande di ricerca relative a:

1. Quali sono le caratteristiche linguistiche che rendono una frase più o meno accettabile o complessa;
2. Quanto un Neural Language Model riesce a modellare giudizi umani di accettabilità e complessità linguistica;
3. Come varia, nei diversi layer, la conoscenza di un Neural Language Model dopo il fine tuning sui due compiti linguistici;

Nel tentativo di rispondere a questi interrogativi si proverà a comprendere, anche, quanto questi due compiti sono simili tra loro.

Nello specifico, nel terzo capitolo, oltre a una breve descrizione dei task di accettabilità e complessità linguistica e dei corpora utilizzati, si illustreranno le caratteristiche linguistiche maggiormente correlate con i due compiti, analizzando anche in che misura queste siano comuni.

Successivamente, nel quarto capitolo, verranno descritti esperimenti di predizione dei punteggi di valutazione di accettabilità e complessità delle frasi. Più nello specifico, verrà illustrata la capacità, sia di un regressore lineare (un *Support Vector Regression* con kernel *linear*) che di BERT, nel predire i punteggi su una scala Likert da 1 a 7 di accettabilità e complessità, assegnati alle 672 frasi del dataset rilasciato per *AcCompl-it* (EVALITA 2020). In questo capitolo, inoltre, si è cercherà di capire quanto la presenza di feature linguistiche esplicite influenzi le performance del modello di regressione.

Infine, nel quinto capitolo, verrà svolta un'analisi approfondita dei modelli precedentemente elaborati (ossia: i due direttamente testati sui due compiti linguistici

di *accettabilità* e *complessità* e i due che sfruttano tecniche di *transfer learning*). Più in particolare, verranno estratte diverse rappresentazioni delle frasi del dataset IUDT da ciascuno strato di ogni modello e poi valutata la similarità cosenica tra queste. Successivamente, effettuando 82 *probing task*, verrà descritta una profilazione linguistica dei diversi modelli neurali, soffermandosi su come cambiano le loro competenze lungo i diversi layer.

Ancora una volta, si cercheranno di individuare somiglianze e differenze sia nelle rappresentazioni che nelle capacità acquisite successivamente al *fine tuning* sui diversi compiti linguistici.

2. Stato dell'arte sui modelli di rappresentazione del linguaggio

Questo capitolo offre un quadro storico sui diversi modelli computazionali semantici che si sono succeduti nel tempo fino ad arrivare agli attuali modelli neurali, di cui si illustreranno studi correlati al progetto di tesi.

2.1 Prospettiva storica

Già a partire dalla metà del secolo scorso, si è cercato di sviluppare dei modelli computazionali che fossero in grado di catturare il significato. In questo contesto è nata la *semantica distribuzionale*, ovvero una famiglia di modelli computazionali che rappresentano il significato come vettori che codificano informazioni estratte dal contesto, in particolare dal modo in cui le parole co-occorrono. I modelli di questa famiglia provano a catturare le proprietà semantiche di espressioni linguistiche in termini della loro distribuzione statistica nel contesto linguistico. In questo modo, le rappresentazioni semantiche sono date da vettori (*embedding*) le cui componenti rappresentano le co-occorrenze del termine lessicale nel contesto linguistico.

Alla base della semantica distribuzionale, vi è la cosiddetta *ipotesi distribuzionale*, secondo cui due parole sono tanto più semanticamente simili quanto più tendono a ricorrere in contesti simili (Miller e Charles, 1991). Rappresentando quindi i termini lessicali come vettori, questi possono essere posti in uno spazio vettoriale (o semantico) in cui si può calcolare la similarità/vicinanza tramite misure matematiche come il coseno.

Il pioniere dell'ipotesi distribuzionale è ritenuto Zelling S. Harris (Harris, 1954), dato che considerava tale metodologia distribuzionale come l'unico approccio scientifico per lo studio del significato linguistico. Il primo utilizzo esplicito del termine "distributional semantics", invece, si deve a Paul Garvin, che in *Computer participation in linguistic research* scrive:

> *«Distributional semantics is predicted on the assumption that linguistic unites with certain semantic similarities also share certain similarities in the relevant environments […] it may be possible to*

group automatically all those linguist unites which occur in the similarly definable environments, and it is assumed that these automatically produced groupings will be of semantic interest.»

Oltre a esser presenti precursori in campo linguistico ed informatico, vi è anche uno spiccato interesse nelle scienze cognitive. Tra i lavori più influenti troviamo quello di George Miller (Miller, 1967), secondo cui nella nostra mente teniamo sempre traccia delle co-occorrenze dei termini linguistici per formare rappresentazioni di parole che sono radicate nel modo in cui queste sono usate nei diversi contesti.

Ripercorrendo l'evoluzione della semantica distribuzionale, si possono individuare diverse generazioni di modelli motivate dai molteplici cambiamenti nell'ambito della linguistica computazionale, nell'intelligenza artificiale e nelle scienze cognitive.

2.1.1 I modelli count

I primi modelli classici sono nati a partire dalla metà degli anni '90 del secolo scorso e si basano su matrici, estendendo il *vector space model* sviluppato in information retrieval. Questi modelli costruiscono le rappresentazioni distribuzionali usando matrici di co-occorrenza che codificano l'informazione contestuale. Più in particolare, le co-occorrenze tra termini lessicali e contesto linguistico sono estratte dal corpus e la distribuzione dei termini lessicali è rappresentata dalla matrice di co-occorrenza, le cui righe corrispondono ai termini lessicali, le colonne al contesto in cui ricorrono e le entrate le loro co-occorrenze di frequenza oppure una funzione che pesi l'importanza del contesto. Questa matrice di co-occorrenze può essere mappata in una nuova matrice ridotta di dimensioni latenti, in modo da diminuire il rumore statistico delle occorrenze superficiali. Le righe della matrice ridotta corrispondono a vettori impliciti, chiamati *word embeddings*, il cui numero di dimensioni è minore rispetto ai vettori espliciti della matrice iniziale e per questo motivo sono considerati vettori densi. Nei *word embeddings* le dimensioni non corrispondono più ai diversi contesi linguistici, ma a feature semantiche latenti.

Una volta ottenuta la matrice finale si può calcolare la similarità tra i lessemi, semplicemente misurando la similarità tra le diverse righe.

Questa prima classe di modelli classici è stata lo standard fino al 2013 ed è chiamata *modelli count,* in quanto conta letteralmente il numero di volte in cui le parole co-occorrono nei diversi contesti.

Dalla fine degli anni ’90 all’inizio del 2010 è stata condotta molta ricerca basata sull’esplorazione dei diversi tipi di contesto. In un *latent semantic approach* utilizzato in semantica distribuzionale per ridurre le dimensioni della matrice di co-occorrenza, il contesto è considerato a livello di documento. Oggigiorno, invece, la definizione di contesto più utilizzata è quella di *collocazione*, ovvero si utilizzano come contesto le parole collegate o dal punto di vista di una finestra lessicale o da relazioni sintattiche.

2.1.2 I modelli predict

Una seconda generazione di modelli di semantica distribuzionale nasce con l’esplosione del deep learning dell’intelligenza artificiale intorno al 2013. Questi modelli, chiamati *predict*, sono basati sulla nozione di predizione. Mentre nei modelli di tipo *count* viene registrato il numero di occorrenze della parola target nei diversi contesti in maniera esplicita, questa seconda classe di modelli è addestrata a predire i contesti in cui la parola si presenta. Sulla base di queste predizioni, i modelli costruiscono vettori distribuzionali (*embedding*) che salvano implicitamente i contesti. Con i modelli di tipo *predict*, quindi, due parole che co-occorrono con contesti analoghi alla fine avranno embeddings simili, perché vengono predetti contesti simili. La più famosa rete neurale nata con questo tipo di approccio *predict* è Word2Vec (Mikolov, 2013). Questa rete utilizza due tipi di algoritmi per la realizzazione degli embeddings:

1. *Skipgram*;
2. *Cbow* (Continuous Bag Of Words);

Con *skipgram* gli *embeddings* vengono costruiti predicendo le parole che si trovano a sinistra e a destra del target. La dimensione della finestra delle parole da predire è fissata a priori come parametro.

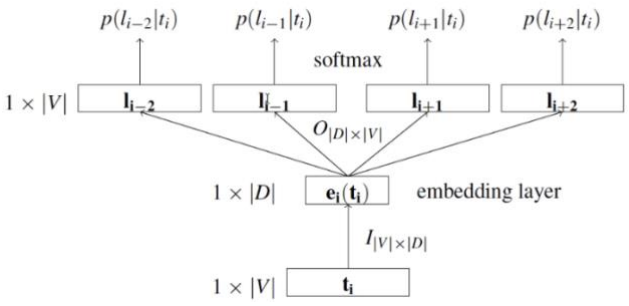


Figura 2.1. Funzionamento di skipgram

Nella figura 2.1. sono rappresentati i layer della rete neurale con algoritmo *skipgram*. Dato un corpus C, ogni parola del vocabolario V viene rappresentata come vettore one-hot in maniera univoca. Il layer in input, quindi, consiste in un insieme di vettori one-hot con V dimensioni che codificano i lessemi del vocabolario. Appena sopra il layer in input, troviamo la matrice I, il cui numero di righe è uguale alla dimensione del vocabolario e il numero di dimensioni corrisponde alla dimensione degli *embeddings* impliciti che si desiderano come risultati. Prima di addestrare la rete, ovviamente i pesi della matrice sono inizializzati in maniera randomica.

L'*embedding layer* computa il dot product tra i pesi della matrice di input e il vettore di input, senza nessuna funzione di attivazione. Infine, il layer di output calcola il dot product tra il vettore di output dell'embedding layer e la matrice di pesi di output $O_{|D|x|V|}$. Applicando poi una funzione softmax si ottiene la probabilità di una parola di apparire nel contesto di t_i. La funzione di loss, partendo da questi vettori probabilistici della softmax, misura la discrepanza tra il vettore risultante e quello corretto, in modo da produrre un segnale di errore e aggiornare i pesi della rete tramite backpropagation. Una volta che la funzione di loss è minimizzata, gli *embeddings* sono codificati tra le righe della matrice I.

Con *cbow*, invece, gli *embeddings* vengono generati predicendo la parola target date una serie di parole contesto.

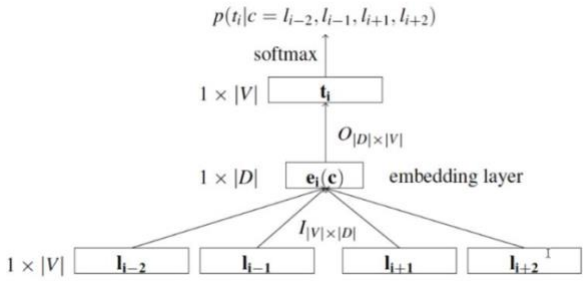


Figura 2.2. Funzionamento di cbow

In questo caso, come illustrato nella figura 2.2, avremo più vettori di input. Nell'*embedding layer* è calcolato il dot product tra i vettori di input e la matrice di input, in modo da ottenere l'*embedding* di ciascun vettore iniziale. Questi *embedding* sono combinati tra loro in modo da ottenere un unico *embedding* delle parole contesto. Successivamente il vettore combinato viene moltiplicato con la matrice di output e viene applicata una softmax. Come con *skipgram*, tramite backpropagation si propaga il segnale di errore e si modificano i pesi delle matrici della rete.

Questo metodo è chiamato *continuous bag of words* perché per combinare i vettori di input in un unico vettore delle parole contesto si utilizza la semplice somma, trattando quindi i termini come "sacco di parole" in cui l'ordine è irrilevante.

Tra il 2014 e il 2016 è nato un dibattito su quale classe di modelli fosse migliore. Nonostante alcuni esperimenti abbiano mostrato che i *prediction model* superano i *count model* in diversi task (Baroni e altri, 2014), per ora i due approcci non differiscono in modo rilevante relativamente al significato che sono in grado di catturare. In ogni caso, è stato dimostrato che, una volta ottimizzati, i *count model* performano meglio dei *predict model* (Omer e altri, 2015), che però diventano competitivi a un costo maggiore, solo se addestrati su un grande quantitativo di dati (Sahlgren e Lenci, 2016). Si tratta comunque di un dibattito ancora aperto.

2.1.3 Limiti di queste prime due classi di modelli

Di seguito verranno illustrati alcuni dei maggiori limiti delle due classi di modelli di semantica distribuzionale presentate finora.

1. Un primo deficit di questi modelli consiste nell'incapacità di catturare le relazioni semantiche che legano le parole e, quindi, le inferenze. Se da una parte sono in grado di riconoscere che termini come *truck* e *vehicle* sono vicini semantici di *car*, non sono invece capaci di stabilire che, nel primo caso, la relazione che lega le due parole è una relazione di co-iponimia, mentre nel secondo di iperonimia. Questo implica che non vengano nemmeno dedotte inferenze tipiche delle due diverse relazioni semantiche, ossia data una frase come "John has a car", non sapendo che *vehicle* è un suo iperonimo, non si può inferire che "John has a vehicle". Riconoscere i vicini semantici, quindi, non è sufficiente per spiegare un aspetto importante della nostra conoscenza semantica delle parole che riguarda le inferenze associate alle relazioni semantiche. Un modello come *cbow* ha un'accuratezza solamente del 38% nel riconoscimento delle inferenze derivanti dall'iperonimia e del 45% nell'individuazione di inferenze come la negazione o la fattività;
2. Un altro aspetto problematico è quello degli antonimi, talvolta chiamato *dilemma dell'antonimia*. Questo tipo di difficoltà era già noto fin dall'inizio della semantica distribuzionale (Miller e Charles, 1991). Miller e Charles,

infatti, riconoscono che un possibile controesempio dell'ipotesi distribuzionale riguardi proprio l'antonimia: molto spesso termini antonimi tra loro, seppur con significato opposto, possono essere liberamente interscambiati negli stessi contesti. Infatti, se un soggetto ha un particolare attributo, questo può essere anche accompagnato dall'aggettivo polare a quell'attributo. Questo comporta che gli antonimi finiranno con l'essere estremamente simili dal punto di vista distribuzionale, ricorrendo negli stessi contesti. Dunque, un aggettivo come *good* avrà una similarità distribuzionale analoga sia nel caso venga confrontato con *bad* (suo antonimo) che con *excellent*, essendo confuso dai modelli distribuzionali;

3. Un altro limite di questi modelli riguarda la polisemia. Infatti, i DSM (Distributional Semantic Models) costruiscono vettori delle parole tipo, tenendo in considerazione tutta la storia distribuzionale dei termini lessicali. Se abbiamo un token polisemico, come ad esempio *play*, questi modelli, analizzando tutti i diversi contesti in cui ricorre, finiranno con il costruire un vettore unico su cui afferiscono i diversi sensi della parola. Questo tipo di problema, in letteratura, è noto come *meaning conflation deficiency*.

   C'è molta ricerca nell'ambito della semantica distribuzionale per creare rappresentazioni *context sensitive* in grado di tener traccia del cambiamento del significato di una parola in diversi contesti. Uno dei contributi più interessanti è quello di Erk e Padó del 2008. Per distinguere i vari significati del verbo *run* in frasi come "the horse runs" e "water runs", propongono di comporre il significato del vettore del verbo con il contesto dato dal soggetto. Ad esempio, nel caso della prima frase, suggeriscono di creare un unico vettore *prototipico* sommando tra loro i vettori relativi alle azioni più tipiche dei cavalli e, successivamente, modificare il vettore di *run* combinandolo con questo vettore prototipo (tramite operazioni come somma o moltiplicazione). In questo modo si potrebbe sia ovviare il limite dei modelli distribuzionali nel trattare la polisemia sia implementare il principio della *cocomposizione* di Pustejovsky, secondo cui quando le parole si compongono tendono a modificarsi l'un l'altra dal punto di vista semantico. Tale soluzione, però, richiede una maggiore elaborazione rispetto all'output dei DSM e, quindi, anche un maggior costo computazionale;

4. Il maggiore collo di bottiglia dei modelli di semantica distribuzionale, però, riguarda la *composizionalità*, ovvero l'idea (che ha origine con Frege) secondo cui il significato di un'espressione complessa è funzione dei significati delle sue parti e di come sono combinate sintatticamente. Questo presuppone che per la costruzione della semantica di un'espressione complessa, sia necessario avere un processo di assegnazione dei significati alle unità di base e poi una operazione semantica che riesca a combinare questi generando il significato finale dell'espressione. Mentre nel caso della semantica formale, quest'operazione è data dall'*applicazione funzionale*[1], nella semantica distribuzionale non è presente un'operazione che combini i vettori in maniera soddisfacente. Nel corso degli anni sono stati proposti diversi modelli per costruire il significato delle espressioni linguistiche complesse in maniera composizionale:
    a. La soluzione più semplice, proposta nel 1997 da Landauer e Dumais, prevede la creazione di una rappresentazione di una struttura complessa mediante la semplice *somma* di vettori. Questo modello, però, risulta insoddisfacente dal punto di vista semantico perché considera le parole come *bag of words*, in quanto, essendo la somma commutativa, vengono ignorati i vincoli di ordinamento delle parole. Inoltre, dal punto di vista matematico, calcolare la somma tra due vettori è assimilabile al farne media e anche questo costituisce un limite a livello teorico perché il significato di un'espressione non è intermedio tra la semantica delle sue parti. Infine, un ulteriore problema che vale la pena di menzionare riguarda l'incapacità del modello additivo di tenere traccia delle diverse modulazioni dei significati delle parole. Tuttavia, nonostante i molteplici limiti a livello linguistico, produce prestazioni migliori rispetto ai metodi sviluppati fino ad oggi (Wieting e altri, 2016);
    b. Un'alternativa degna di nota è quella dei *sentence embeddings*. In questo caso, l'idea è quella di utilizzare reti neurali capaci di costruire un unico vettore che codifica un'intera frase. I tipi di *neural network* utilizzati

[1] Operazione secondo cui alcuni termini di un'espressione complessa sono *funzioni*, mentre altri sono *argomenti* e applicando le *funzioni* agli *argomenti* si costruisce la semantica dell'espressione complessa.

inizialmente erano le LSTM (long short-term memory), ovvero una nuova generazione di *recurrent neural network*, che una volta addestrate sia su task supervisionati che non (ad esempio come autoencoder o per la traduzione automatica), sono in grado di costruire una rappresentazione interna che codifica aspetti rilevanti del significato della frase. La più importante architettura neurale in grado di generare *sentence embedding* è quella dei modelli *encoder-decoder* (anche noti come *seq2seq*). Questi modelli sono chiamati così perché costituiti da due parti, per l'appunto, un *encoder* e un *decoder*. L'*encoder* processa un'intera frase codificandola in un unico vettore che poi è passato come input al *decoder*. Quest'ultimo, a sua volta, genera la sequenza di output richiesta per la risoluzione del task preso in considerazione. Il *sentence embedding* è, quindi, il risultato della rappresentazione interna della rete al fine dello svolgimento del compito.

Il limite più evidente del modello *seq2seq* consiste nel fatto che l'intera informazione estrapolata dal testo da tradurre è concentrata nello stato finale dell'*encoder*. Questa sintesi in una unica grandezza costituisce il più rilevante collo di bottiglia del sistema, specie per testi lunghi. Infatti, le performance del modello degradano progressivamente con l'aumentare della lunghezza del testo da analizzare. Per risolvere tale limite, è necessario introdurre un meccanismo di *attention* che consente di concentrarsi sulle parti rilevanti della sequenza in input.

Anche dal punto di vista linguistico-teorico si riscontrano alcuni problemi nei *sentence embedding*, come ad esempio l'incapacità di discriminare tra diverse realizzazioni sintattiche dei ruoli semantici (Zhu e altri, 2018). Una frase come "Lilly loves Imogen" erroneamente viene riconosciuta come più simile a "Imogen loves Lilly" piuttosto che alla sua controparte passiva.

Inoltre, come dimostrano Wieting e altri nel 2019, i *sentence encoders* non addestrati che usano *embeddings* pre-trainati performano allo stesso modo di quelli addestrati. Pertanto, la maggior parte del potere dei sistemi di NLP moderni non deriva dai *sentence encoder*, ma dalla qualità degli *embeddings* di partenza. Dunque, il valore aggiunto di

questi modelli, al fine di catturare la semantica di una frase, è molto limitato.

2.1.4 I modelli distribuzionali contestualizzati

A partire dai 2019, nasce una nuova generazione di modelli distribuzionali. Mentre i modelli di semantica distribuzionale visti finora producono embedding statici, assegnando un vettore ad ogni parola tipo (creando quindi un unico *embedding* per i diversi significati di termini polisemici), i nuovi modelli – noti come *contestualizzati* – assegnano una rappresentazione distinta a ciascun token del testo. In questo modo, ciascuna parola in un diverso contesto ha una diversa rappresentazione.
Il principale modello distribuzionale contestualizzato utilizzato oggigiorno in NLP è BERT (Bidirectional Encoder Representations from Transformers), un *language model* basato sui Transformer sviluppato da Google nel 2018 (Devlin e altri, 2018).

2.1.5 Transformer

Prima di illustrare il funzionamento di BERT, però, al fine di comprenderlo meglio, è necessario fare un passo indietro analizzando l'architettura dei Transformer. Questi modelli (proposti per la prima volta nel paper *Attention is All You Need*, 2018) costituiscono una vera e propria svolta nel campo dell'intelligenza artificiale perché, utilizzando meccanismi di *attenzione*[2], permettono di ottenere performance superiori in task di traduzione automatica, sono più parallelizzabili e richiedono un minore tempo di addestramento.
Come mostrato nella figura 2.3, l'architettura dei Transformer è composta da due componenti: uno stack di *encoder* e uno di *decoder*.

[2] Tecnica che permette al modello di focalizzarsi sulle parti rilevanti della sequenza in input, introdotta per superare i limiti di architetture seq2seq.

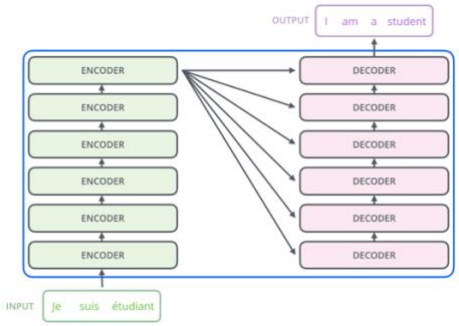


Figura 2.3. Architettura dei Transformer

Gli encoder hanno tutti una stessa struttura suddivisa in due sotto-layer: una *feed forward neural network* e uno strato di *self-attention* (Figura 2.4). L'input dell'*encoder* viene prima passato al *self-attention layer*, che genera un output che successivamente viene dato in pasto alla *feed-forward neural network*.

Anche il *decoder* ha entrambi questi layer, ma tra loro è presente un *encoder-decoder attention layer* che aiuta il *decoder* a focalizzarsi sulle parti rilevanti della sequenza in input.

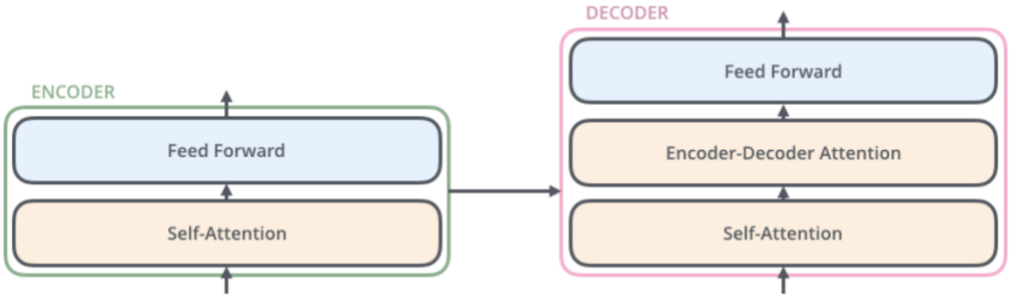


Figura 2.4. Struttura interna degli *encoder* e dei *decoder*

Il primo *encoder* dello stack degli *encoder* prende come input gli *embedding* (ciascuno di dimensione 512) delle parole della frase da analizzare. Gli altri *encoder*, invece, ricevono in ingresso l'output dell'*encoder* immediatamente precedente. La dimensione della lista di vettori in input è un iperparametro, impostato tipicamente della stessa lunghezza della frase più lunga del dataset di training.

Una volta generati gli *embedding*, questi vengono dati in pasto al primo *encoder*, seguendo individualmente il proprio path, in modo parallelo.

Quando l'*embedding* giunge al *self-attention layer*, questo è in grado di individuare tra le altre posizioni della sequenza in input indizi che possono condurre a una migliore codifica della parola. Ad esempio, se viene processato il token "it" (della frase "The

animal didn't cross the street because it was too tired"), la *self attention* permette di associarlo ad "animal".

Il primo step per il calcolo della *self attention* consiste nel creare tre vettori (*query vector*, *key vector*, *value vector*) per ciascun vettore dato in input all'*encoder*. Questi tre vettori sono generati dal prodotto tra l'*embedding* iniziale e tre matrici che sono addestrate durante il processo di training. Il *query vector*, *key vector* e *value vector* hanno dimensioni inferiori rispetto agli *embedding* originari: infatti, mentre l'input del primo *encoder* ha dimensione 512, questi vettori, invece, hanno dimensionalità 64. In figura 2.5 si nota come il vettore q_1 è ottenuto dal prodotto tra l'*embedding* iniziale x_1 e la matrice W^Q.

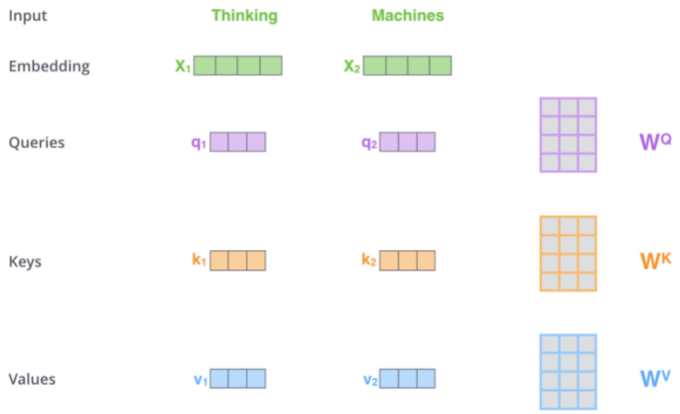


Figura 2.5. Costruzione del query vector, key vector, value vector

Il secondo step per il calcolo della *self-attention* riguarda la computazione di uno score. Questo punteggio è ottenuto dal confronto di ogni termine con il resto della frase, per determinare quanto focalizzarsi sulle altre parti nella codifica di quel termine. In termini matematici, il calcolo è dato dal *dot product* del *query vector* con il *key vector* della parola a cui stiamo assegnando un punteggio. Quindi, se stiamo processando la *self-attention* della parola in posizione #1, il primo score è dato dal *dot product* tra q_1 e k_1 e il secondo dal *dot product* tra q_1 e k_2 (come illustrato in figura 2.6).

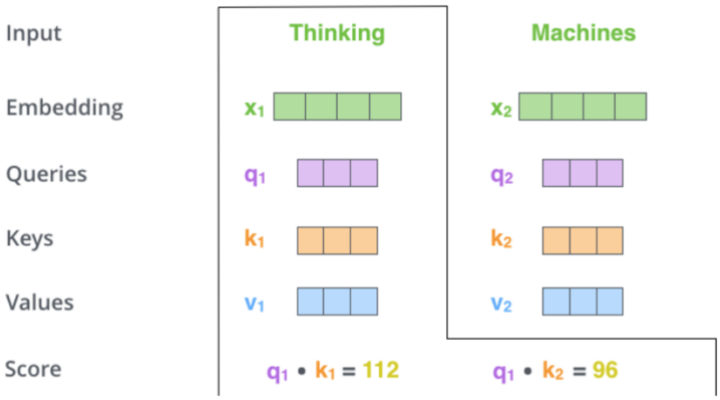


Figura 2.6. Secondo step del calcolo della *self-attention*

Il terzo e il quarto step (figura 2.7) consistono nel dividere gli score per 8 (la radice quadrata della dimensione del *key vector*) e passare i risultati a una *softmax* al fine di normalizzare i punteggi in modo che siano positivi e la loro somma sia 1.

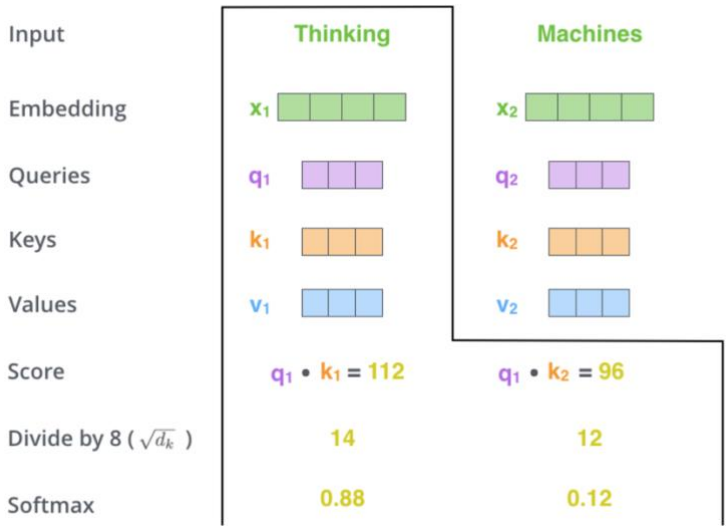


Figura 2.7. Terzo e quarto step del calcolo della *self-attention*

Il quinto step consiste nel moltiplicare ciascun *value vector* per il punteggio della softmax, in modo da mantenere intatti i valori delle parole su cui vogliamo focalizzarci e attutire, invece, le parole meno rilevanti (moltiplicandole per numeri più piccoli).
Infine, il sesto step prevede la somma dei *value vector* pesati per ottenere l'output del *self-attention* layer per ciascuna posizione (in figura 2.8 è visibile l'output della prima posizione).

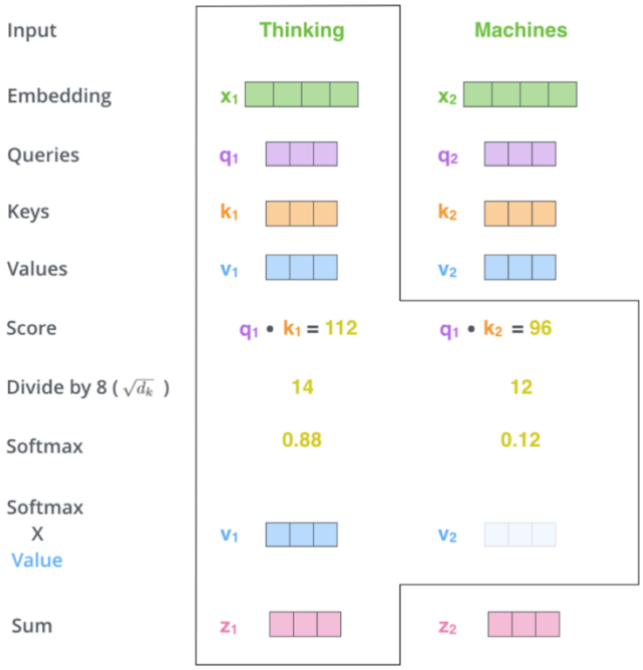


Figura 2.8. Output del *self-attention* layer

A questo punto, il vettore risultante può essere dato in pasto alla *feed-forward neural network.*

Tendenzialmente, le operazioni illustrate finora vengono eseguite in forma matriciale per velocizzare il processo di calcolo della *self-attention* e renderlo parallelizzabile. Più in particolare, nel primo step, anziché creare tre vettori, vengono calcolate tre matrici (*query*, *key* e *value*), unendo gli *embedding* in input in una matrice X e moltiplicandola per le matrici pesate addestrate precedentemente W^Q, W^K e W^V (figura 2.9).

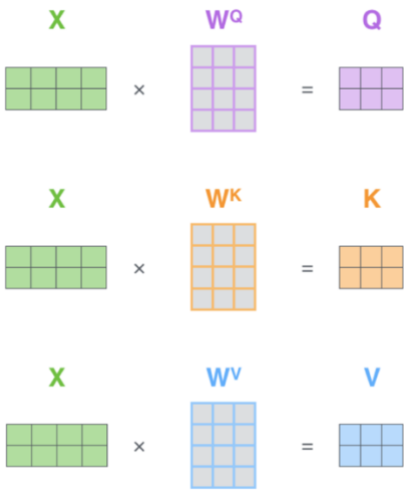


Figura 2.9. Primo step del calcolo matriciale della *self-attention*

Infine, lavorando con le matrici, gli step da due a sei possono essere riassunti in un'unica formula (figura 2.10).

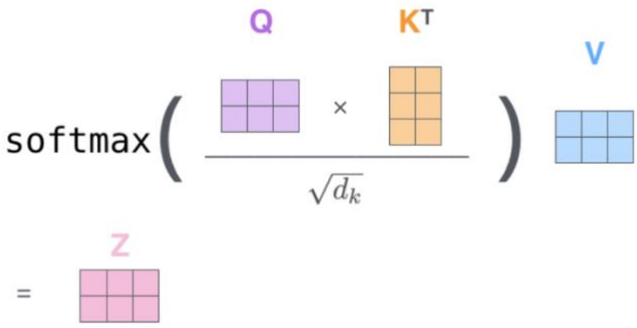


Figura 2.10. Calcolo matriciale della *self-attention*

Il paper *Attention is all you need* (Vaswani, 2017), inoltre, rifinisce il *self-attention* layer aggiungendo un meccanismo chiamato *multi-headed attention*. Questo migliora le performance in due modi:

1. Espande la capacità del modello di focalizzarsi su diverse posizioni;
2. Dà all'*attention layer* diversi sottospazi di rappresentazione (Figura 2.11). Con la *multi-headed attention*, infatti, abbiamo molteplici insiemi di matrici Query/Key/Value. Più in particolare, dato che i Transformer usano otto *attention heads*, alla fine ci saranno otto insiemi di matrici per ciascun *encoder*/*decoder*. Ciascuna di queste, inizialmente, è inizializzata randomicamente, solo successivamente, dopo il training, è utilizzata per proiettare l'i*nput embedding* (o i vettori degli *encoder*/*decoder* sottostanti) in un diverso sottospazio di rappresentazione;

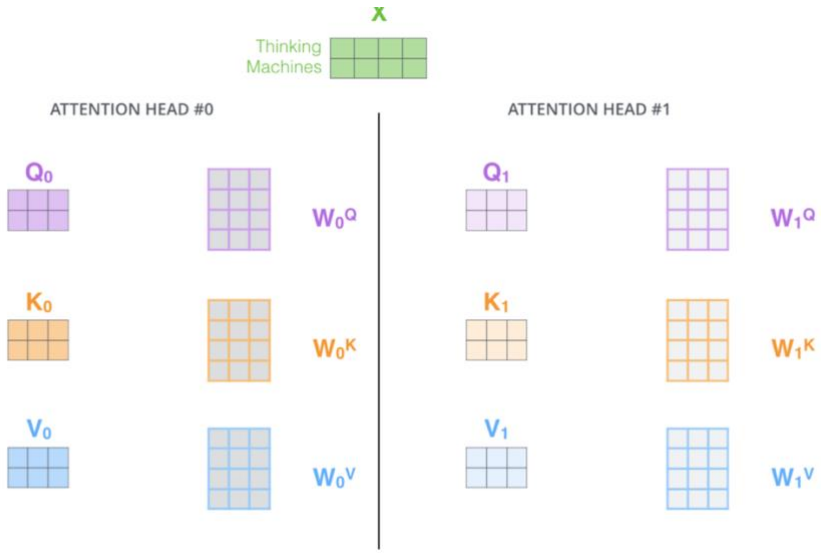
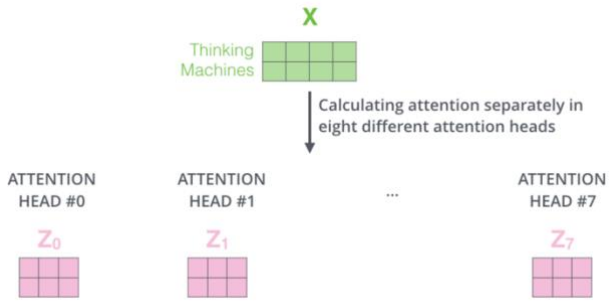


Figura 2.11. *Multi-headed attention*

Facendo lo stesso calcolo della *self-attention* descritto precedentemente con otto differenti insiemi di matrici Q/K/V, si otterranno otto diverse matrici Z (Figura 2.12).



2.12. Matrici Z nella *multi-headed attention*

Per ottenere un'unica matrice Z da passare al *feed-forward layer*, si condensano le otto matrici Z ottenute per ciascuna *head*, concatenandole tra loro e moltiplicandole per un'ulteriore matrice pesata W^O addestrata insieme al modello (figura 2.13).

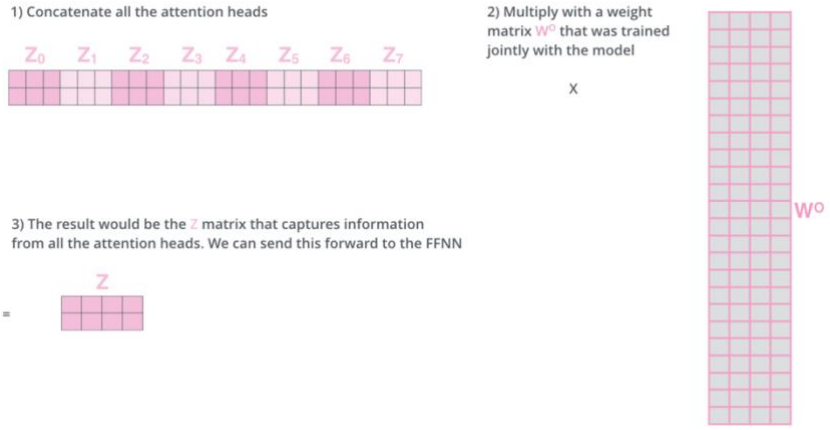


Figura 2.13. Output da passare al *feed-forward layer*

Un'altra caratteristica importante dei Transformer è il *positional* encoding, ossia il modo in cui viene tenuto traccia dell'ordine delle parole della sequenza in input. A questo scopo, viene aggiunto un vettore a ciascun *input embedding*. Questi vettori seguono uno specifico pattern imparato dal modello, in modo da aiutarlo nel determinare la posizione di ciascuna parola o la distanza tra le diverse parole nella sequenza.

Inoltre, per completare la descrizione dell'architettura, è necessario aggiungere che dopo ciascun sotto-layer degli *encoder* o dei *decoder* è presente uno step di normalizzazione. In figura 2.14 si mostra come risulterebbe un Transformer con due *encoder* e decoder:

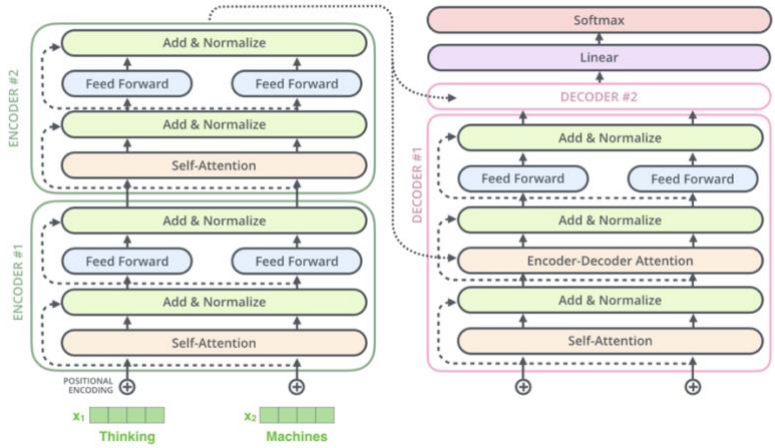


Figura 2.14. Architettura completa di un Transformer

Una volta conclusa la fase di *encoding*, l'output dell'ultimo *encoder* è trasformato in un insieme di vettori *attention* K e V. Questi sono utilizzati nell'*encoder-decoder attention layer* di ciascun *decoder* in modo da riuscire a focalizzarsi sulle parole più importanti della sequenza in input. Ciascuno step della fase di decodifica produce un elemento della sequenza di output finché non si raggiunge un simbolo che indica al *decoder* la fine della frase. Inoltre, l'output di ciascuno step è dato in pasto al *decoder* più in basso e, come accadeva con l'input degli *encoder*, è aggiunta una codifica posizionale a ciascuna parola dell'*input* del *decoder*. L'intero processo è riassunto nella figura 2.15.

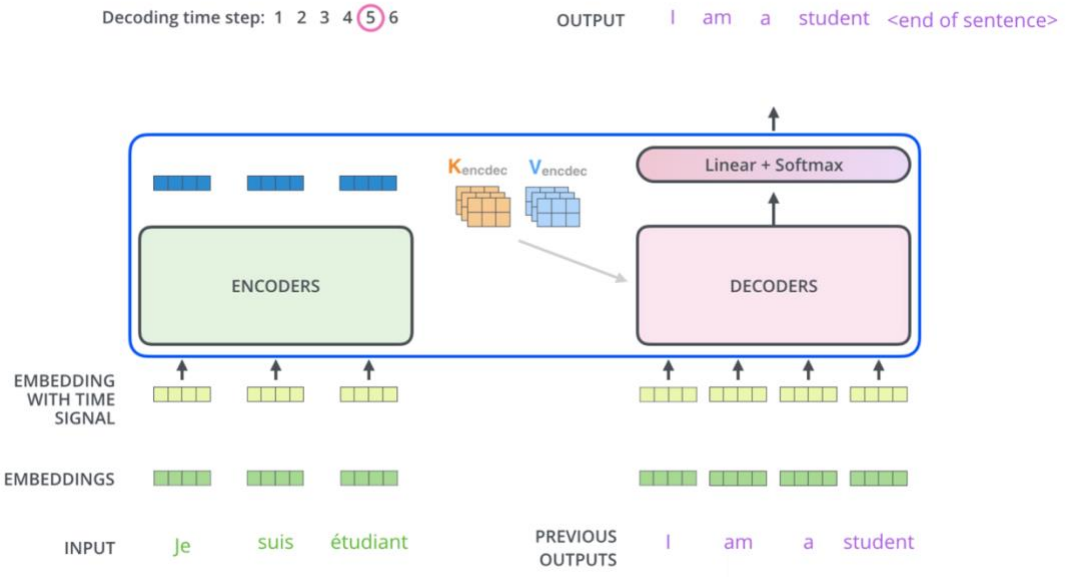


Figura 2.15. Fase di decodifica

Rispetto all'*encoder*, il *self-attention layer* del *decoder* opera in modo leggermente diverso. Nel *decoder*, questo layer può accedere solo alle posizioni precedenti della sequenza di output. Le posizioni successive, infatti, sono mascherate prima dello step del calcolo della softmax per l'ottenimento della *self-attention*.
Per quanto riguarda il funzionamento dell'*encoder-decoder attention layer*, invece, si può dire che funzioni similmente alla *multi-headed self-attention*, ad eccezione del fatto che crea le matrici Q dal layer sottostante e prende le matrici K e V dall'output della pila degli *encoder*.

Una volta generato l'output della pila dei *decoder*, questo è dato in pasto a un *linear layer* e una *softmax*. Il *linear layer* è una semplice rete neurale *fully connected* che proietta il vettore prodotto dallo stack dei decoder in un molto più grande vettore chiamato *logits vector*. Assumendo che il nostro modello conosca 10 mila parole diverse, che ha imparato dal suo training dataset, il *logits vector* avrà 10 mila celle, ciascuna corrispondente allo score di ciascuna parola del vocabolario.
Il *softmax layer* successivamente converte questi punteggi in probabilità (in modo che siano tutte positive e che la loro somma sia uno). Viene scelta la cella con più alta probabilità e la parola associata a questa è prodotta come output a quello step.

Il procedimento appena descritto viene seguito anche durante la fase di training. Durante questa fase, però, trattandosi di apprendimento supervisionato, l'output prodotto viene comparato con l'output corretto, in modo poi da modificare i pesi del modello tramite *back propagation*.

2.1.6 BERT

BERT (Bi-Directional Encoder Representations from Transformers) è una rete neurale molto complessa che si basa su una pila di *encoder* Transformers. La sua architettura tendenzialmente si presenta con dodici *encoder* layer (anche chiamati *transformer block*) nella versione BERT base e con ventiquattro *encoder* layer nella versione BERT large (figura 2.16). Inoltre, entrambe le versioni contengono ampie *feedforward-network*, rispettivamente di 768 e 1024 unità nascoste, e più *attention head* (12 in BERT base e 16 in BERT large).

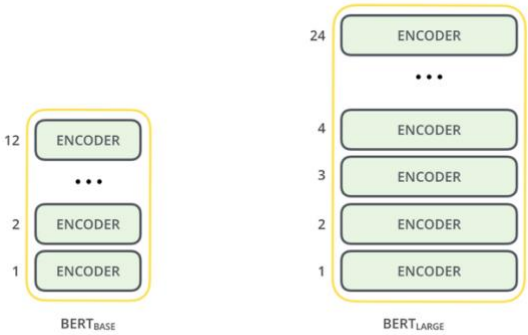


Figura 2.16. BERT base e BERT large

Come nel caso dei Transformer illustrati finora, BERT prende in input la sequenza degli embedding delle parole della frase, preceduta da un token chiamato [CLS]. Ciascun *encoder* applica la *self-attention* e passa il risultato a una *feed-forward network*, il cui output diviene l'input dell'*encoder successivo* (figura 2.17).

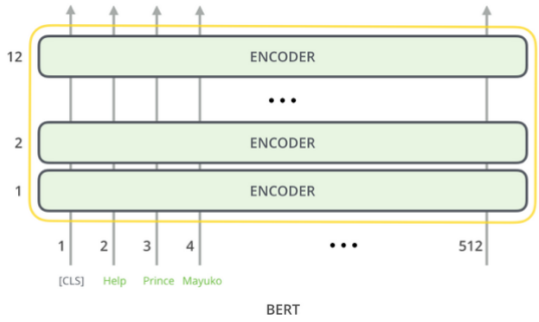


Figura 2.17. Funzionamento BERT

Per quanto riguarda l'*output* del modello (figura 2.18), al termine della pila degli encoder, ciascuna posizione restituisce un vettore di 768 unità nascoste (nel caso di BERT base).

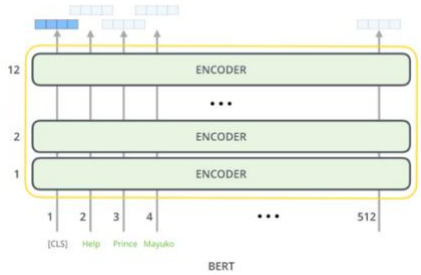


Figura 2.18. Output di BERT

Questo vettore può essere dato in pasto a un classificatore/regressore utile per il raggiungimento dell'obiettivo che ci siamo posti (come *la classificazione* di notizie come *spam/non spam*, la *sentiment analysis* dando giudizi come *positivo*/*negativo*, il *fact-checking* ecc.). Nell'esempio in figura 2.19, si illustra un task di classificazione delle frasi:

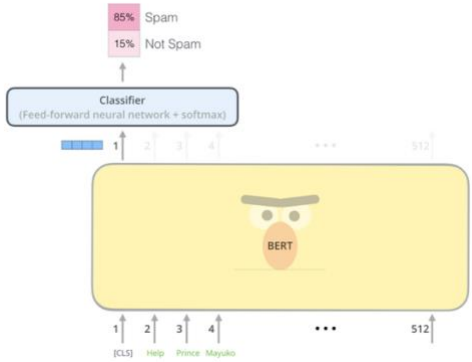


Figura 2.19. Esempio di classificazione con l'output di BERT

Più in generale, ogni specifico task viene risolto pre-trainando BERT al fine di comprendere il linguaggio e successivamente facendo fine-tuning per imparare uno specifico task.

L'obiettivo del *pretraining* è quello di imparare il linguaggio e il contesto, addestrandosi simultaneamente su due task non supervisionati: il *masked language modeling* (MLM) e la *next sentence prediction* (NSP). Per quanto riguarda il *masked language modeling*, BERT prende in input frasi con alcune parole mascherate casualmente e tenta di prevedere la parola "nascosta" (figura 2.20). Questo aiuta BERT a comprendere un contesto bidirezionale all'interno di una frase.

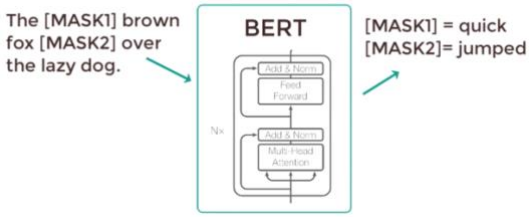


Figura 2.20. Masked language modeling

Nel caso della *next sentence prediction*, invece, BERT prende in input due frasi e determina se la seconda frase è successiva alla prima (figura 2.21). In questo modo, BERT comprende il contesto tra le diverse frasi.

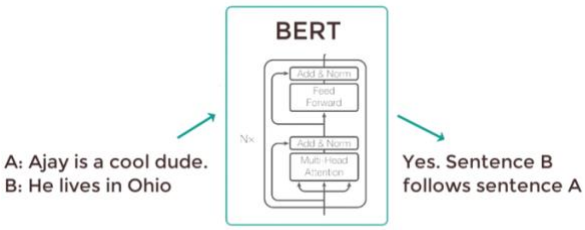


Figura 2.21. Next sentence prediction

Questi due task vengono eseguiti contemporaneamente, infatti, nella fase di *pretraining*, l'input consiste in un insieme di due frasi in cui alcune parole sono mascherate. Ciascuna parola delle due frasi viene successivamente trasformata in un *embedding* E. L'output consiste in C, per quanto riguarda il task di *next sentence prediction* e in $T_1..T_N$ per il task di *masked language modeling* (figura 2.22).
C sarà uguale a 1 se la frase B segue la frase A o a 0 se ciò non avviene. Ciascuna T corrisponde, invece, al vettore di una parola.

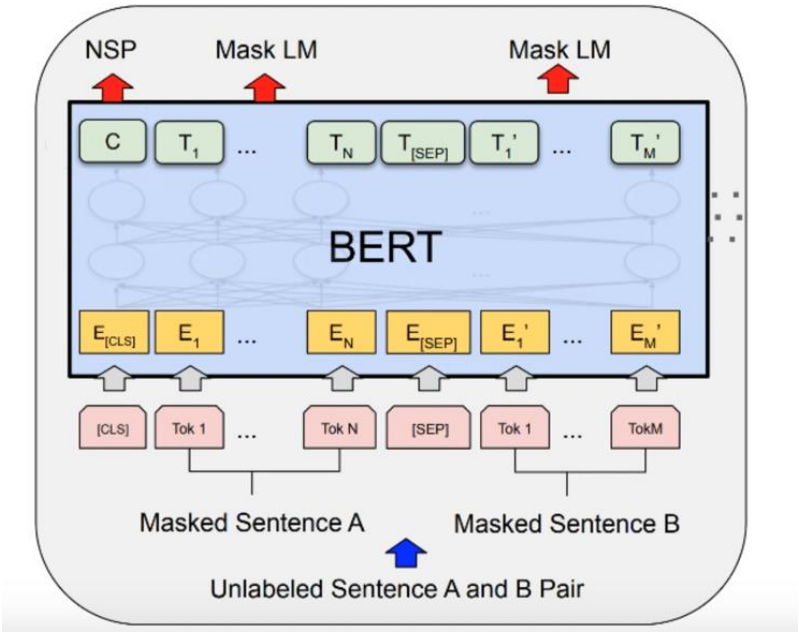


Figura 2.22. Pretraining BERT

Una volta che la fase di *pretraining* è completata, lo step successivo è il fine tuning che consiste nell'addestramento supervisionato necessario per il task che si vuole risolvere. Semplicemente inserendo un layer aggiuntivo di output, si può eseguire il *fine-tuning* del modello e raggiungere performance allo stato dell'arte attuale su un'ampia gamma di compiti, senza modifiche specifiche all'architettura.

Infatti, una grande novità di questo tipo di struttura è proprio che è generica e può adattarsi, tramite *fine-tuning*, a diversi task specifici. Mentre prima (ad esempio con *word2vec*) le rappresentazioni erano prodotte da architetture esclusive per un task, più recentemente i modelli Transformer si sono mostrati direttamente utilizzabili su uno specifico task tramite fine-tuning, rimuovendo la necessità di modificare la loro struttura (Brown e altri, 2020).
Le nuove architetture come BERT, quindi, cercano di imparare molto di più del significato delle parole. Infatti, durante la fase di training, sviluppano un insieme ampio di capacità, che successivamente adattano per conseguire il task desiderato.

Il vantaggio principale resta comunque l'abilità di questi modelli nel trattare la natura *context sensitive* del significato delle parole, tramite lo sviluppo dei cosiddetti *embedding contestualizzati*.

2.1.7 Limiti di BERT

Nonostante le sue grandi potenzialità, questo modello presenta comunque dei limiti nel trattare aspetti importanti del lessico e della semantica. In particolare, una delle sue maggiori difficoltà consiste nel riconoscere le inferenze semantiche.
In un recente studio (Rocchietti e altri, 2021) partendo da un *challenge set* (FANCY: FActivity, Negation, Common-sense, hYpernymy), costituito da 4000 coppie di frasi inglesi annotate con le loro relazioni di inferenza, è stato valutato in che misura BERT sia in grado di riconoscerle, se precedentemente *fine-tuned* su altro dataset di inferenze noto come MNLI (Williams e altri, 2018).
Come illustrato in figura 2.23, lo stesso modello *fine-tuned* e testato su MNLI ottiene un'accuratezza del 90%, mentre su FANCY ottiene accuratezze minori.

RoBERTa	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
Factivity	0.75	0.75	0.79	0.75
Negation	0.82	0.77	0.75	0.76
Common Sense	0.68	0.67	0.68	0.67
Hypernymy	0.60	0.61	0.68	0.61
MNLI	0.90			

Figura 2.23. Accuratezza nel riconoscimento delle inferenze da parte di BERT

Se da una parte, è vero che questi modelli imparano molto nel *training*, dall'altra si nota come abbiano comunque ricadute nel riconoscere aspetti importanti del significato.
Più nello specifico, vediamo come spesso vengano confuse la contraddizione (C) o la negazione (N) con l'entailment (E) (figura 2.24).

Premise	Hypothesis	Gold	Pred.
The man was born in 1950.	The man was 18 in 1968.	E	C
No arrow hit the target.	Not all arrows hit the target	C	E
Jackie believes that Twin Peaks is the best tv show ever	Twin Peaks is the best tv show ever	N	E
All seagulls fly	All birds fly	N	E

Figura 2.24. Errori tipici nel riconoscimento delle inferenze

Un altro aspetto problematico di BERT riguarda il grande quantitativo di dati che richiede per essere addestrato. Anche se in alcuni casi questo dispendio è compensato da performance migliori, in altri, invece, non è così, specialmente se proviamo a utilizzare questi modelli per task *out-of-context*. Infatti, in questi casi le performance ottenute sono assimilabili o si discostano poco rispetto a quelle degli *embedding statici*.

Alla luce di quanto detto, mentre Landauer e Dumais nel 1997 si chiedevano come fanno le persone a sapere quanto sanno con la poca informazione che hanno[3], oggigiorno, invece, ci si domanda come fanno i modelli di semantica distribuzionale a sapere così poco sul significato con un così grande quantitativo di testi che processano.
Probabilmente questo problema deriva dal fatto che ci sono aspetti del significato che non sono informazioni distribuzionali. Le persone, infatti, imparano anche da esperienze multimodali, contesti interattivi o da insegnamenti espliciti. Questo genere di fattori sfugge a questo tipo di modelli, che quindi, ancora non riescono a catturare bene tutti gli aspetti semantici.

[3] Problema noto come *Plato's Problem*.

Nonostante ciò, attualmente i modelli di semantica distribuzionale contestualizzati risultano comunque tra i migliori modelli allo stato dell'arte, mostrando performance notevoli su molteplici task.

2.2 Profilazione linguistica di un modello neurale del linguaggio

Durante gli ultimi anni, i modelli neurali del linguaggio (come BERT) sono diventati centrali nel campo del Natural Language Processing, superando risultati allo stato dell'arte in molti compiti. Tuttavia, l'introduzione di questi sistemi ha avuto come costo l'interpretabilità. Lavori recenti hanno iniziato a studiare questi modelli al fine di capire se codificano fenomeni linguistici anche senza essere progettati esplicitamente per imparare queste proprietà (Marvin e Linzen, 2018; Goldberg, 2019; Warstadt e altri, 2019). La maggior parte di questi lavori si focalizza sull'analisi e l'interpretazione dei meccanismi di attenzione (Tang e altri, 2018; Jain e Wallace, 2019; Clark e altri, 2019) e sulla definizione di modelli di *probing* addestrati per predire semplici proprietà linguistiche da rappresentazioni non supervisionate.

I modelli di *probing* addestrati su diverse rappresentazioni contestuali forniscono evidenze che questi modelli sono in grado di catturare un'ampia gamma di fenomeni linguistici (Adi e altri, 2016; Perone e altri, 2018; Tenney e altri, 2019b) e persino di organizzare l'informazione in modo gerarchico (Belinkov e altri, 2017; Lin e altri, 2019; Jawahar e altri, 2019). Tuttavia, il modo in cui la conoscenza ha impatto sulle decisioni che prendono quando risolvono un task di base è stata meno studiata.

A partire dal 2019, Jawahar e altri hanno individuato che le rappresentazioni imparate nei primi layer di BERT solitamente catturano meglio le *feature* di superficie, mentre gli *embedding* degli ultimi layer riescono a codificare in modo migliore le proprietà sintattiche e semantiche. Utilizzando un insieme di probing task, Tenney e altri (2019a) hanno scoperto che la conoscenza linguistica codificata da BERT lungo i suoi layer segue la tradizionale pipeline di NLP: *POS tagging*, *parsing*, *NER*, *ruoli semantici* e infine *coreferenze*. Liu e altri (2019), invece, hanno quantificato le differenze tra diversi modelli, mostrando che i layer più alti delle RNN (come ELMo)

sono più specifici per il task (e, quindi, meno generali), mentre i layer dei Transformer (come BERT) non mostrano incrementi nel task specifico.

Tra gli articoli più recenti di questo ambito, inoltre, si colloca un lavoro di Miaschi (2020), che studia le proprietà linguistiche codificate da BERT prima e dopo il *fine-tuning* su uno specifico task a valle, ossia l'identificazione della lingua madre (NLI, *Native Language Identification*). Più in particolare, questo studio si propone di rispondere a tre quesiti di ricerca: 1) quali proprietà linguistiche sono già codificate nella versione preaddestrata di BERT e come sono disposte nei suoi 12 layer, 2) come la conoscenza di queste proprietà è modificata dopo il processo di *fine-tuning*, 3) se la conoscenza implicita del modello ha effetti sulla capacità di risolvere uno specifico task a valle.

Per rispondere a questi interrogativi e, quindi, testare la conoscenza linguistica codificata in BERT, sono stati eseguiti 68 *probing task*, ciascuno corrispondente a una feature distinta che cattura il lessico e le proprietà morfosintattiche e sintattiche di una frase. In primo luogo, è stato effettuato il *probing* dell'informazione imparata in una versione pre-addestrata di BERT (BERT-based, *cased*), usando frasi *gold* annotate in accordo al framework delle Universal Dependencies (Nivre e altri, 2016). In particolare, è stato definito un modello di *probing* che usa le rappresentazioni contestuali di BERT per ciascuna frase del dataset e predice il valore reale di ciascuna feature linguistica lungo i layer interni. In secondo luogo, sono state investigate variazioni nell'informazione linguistica codificata tra il modello preaddestrato e dieci modelli *fine-tuned*, ottenuti su diversi task binari di *native language identification* (NLI). Infine, è stato analizzato come la competenza linguistica contenuta nel modello abbia effetto nel risolvere task di NLI. Tutti questi esperimenti sono stati condotti su due dataset: la treebank UD (versione 2.4) per l'inglese (per testare l'informazione linguistica precedente e successiva al processo di *fine-tuning*) e un dataset usato per il task di NLI. Seguendo l'approccio di *probing* definito da Conneau e altri (2018), che ha lo scopo di catturare informazioni linguistiche da rappresentazioni imparate dal modello del linguaggio naturale, ciascun probing task ha riguardato la predizione del valore di una specifica feature linguistica estratta automaticamente dalle frasi parsate dei due dataset. Le feature utilizzate vanno da informazioni *raw* a livelli morfosintattici e sintattici e possono essere categorizzati in 9 gruppi corrispondenti a diversi fenomeni linguistici (figura 2.25).

Level of Annotation	Linguistic Feature	Label
	Raw Text Properties (*RawText*)	
Raw Text	Sentence Length	sent_length
	Word Length	char_per_tok
	Vocabulary Richness (*Vocabulary*)	
Vocabulary	Type/Token Ratio for words and lemmas	ttr_form, ttr_lemma
	Morphosyntactic information (*POS*)	
	Distibution of UD and language–specific POS	upos_dist_*, xpos_dist_*
POS tagging	Lexical density	lexical_density
	Inflectional morphology (*VerbInflection*)	
	Inflectional morphology of lexical verbs and auxiliaries	xpos_VB-VBD-VBP-VBZ, aux_*
	Verbal Predicate Structure (*VerbPredicate*)	
	Distribution of verbal heads and verbal roots	verbal_head_dist, verbal_root_perc
	Verb arity and distribution of verbs by arity	avg_verb_edges, verbal_arity_*
	Global and Local Parsed Tree Structures (*TreeStructure*)	
	Depth of the whole syntactic tree	parse_depth
	Average length of dependency links and of the longest link	avg_links_len, max_links_len
	Average length of prepositional chains and distribution by depth	avg_prep_chain_len, prep_dist_*
Dependency Parsing	Clause length	avg_token_per_clause
	Order of elements (*Order*)	
	Relative order of subject and object	subj_pre, obj_post
	Syntactic Relations (*SyntacticDep*)	
	Distribution of dependency relations	dep_dist_*
	Use of Subordination (*Subord*)	
	Distribution of subordinate and principal clauses	principal_prop_dist, subordinate_prop_dist
	Average length of subordination chains and distribution by depth	avg_subord_chain_len, subordinate_dist_1
	Relative order of subordinate clauses	subordinate_post

Figura 2.25. Feature linguistiche utilizzate nei *probing task*

Mentre per l'estrazione degli *embedding contestuali* e il processo di *fine-tuning* per il task di NLI è stata usata una versione inglese pre-addestrata di BERT (BERT-base cased, 12 layer, 768 unità nascoste), per il *probing* è stata usata una *LinearSVR* (*linear support vector regression*) che cerca di predire il valore reale di ciascuna feature linguistica data la rappresentazione della frase imparata da BERT per ciascuno dei suoi layer.

Per capire quali fenomeni linguistici sono codificati nella versione preaddestrata di BERT, è stata rappresentata ciascuna frase del dataset UD utilizzando il suo sentence embedding corrispondente e successivamente, per ciascuna rappresentazione sono stati eseguiti i 68 *probing task*, utilizzando il modello *LinearSVR*. Dato che la maggior parte delle feature di *probing* è molto correlata con la lunghezza della frase, sono stati comparati i risultati del *probing* con quelli di una baseline, calcolata misurando la correlazione di Spearman tra la lunghezza di ciascuna frase e il corrisponde valore di probing. La valutazione del probing è stata effettuata con un 5-*fold cross validation* e usando la correlazione di Spearman tra le etichette predette e quelle *gold* come metrica di valutazione.

Analizzando come la competenza linguistica cambia lungo i layer (figura 2.26), si può notare che la media degli score di ciascun layer è più bassa negli ultimi layer per tutti i gruppi distinti.

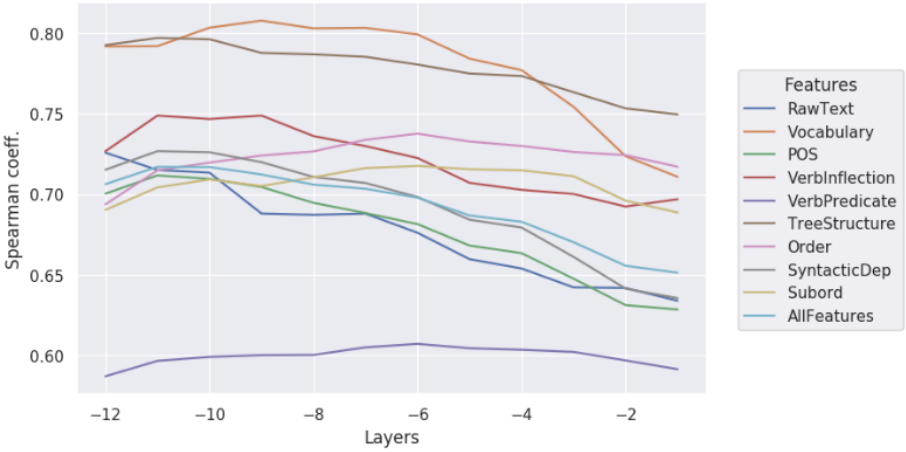


Figura 2.26. Media degli score di Spearman per ciascun layer di BERT preaddestrato

Tuttavia, ci sono differenze tra i gruppi considerati: le competenze delle feature *raw* (*RawText*) e la distribuzione delle POS si perdono nei primi layer (entro il layer -10), mentre la conoscenza circa l'ordine del soggetto/oggetto rispetto al verbo, l'uso della subordinata e tutte le feature correlate alla struttura del predicato verbale sono acquisite nei layer intermedi. Se si considera come cambia la conoscenza di ciascuna feature lungo i layer (figura 2.27), si osserva che non tutte le feature che appartengono allo stesso gruppo hanno un comportamento omogeneo. Questo è il caso, ad esempio, di due feature incluse nel gruppo *RawText*: la lunghezza delle parole (*char_per_tok*) raggiunge punteggi un po' più bassi lungo tutti i layer rispetto alla feature *sent_length*. Allo stesso modo, la conoscenza delle POS differisce quando considerata a diversi livelli di specificità. Ad esempio, tra le ampie categorie dei verbi e dei nomi, sono ottenute predizioni peggiori per classi sotto-specificate dei verbi basate sul tempo, persona o modo (ad esempio *xpos_dist_VBN*) e per i nomi flessi al singolare e al plurale (*NN, NNS*).

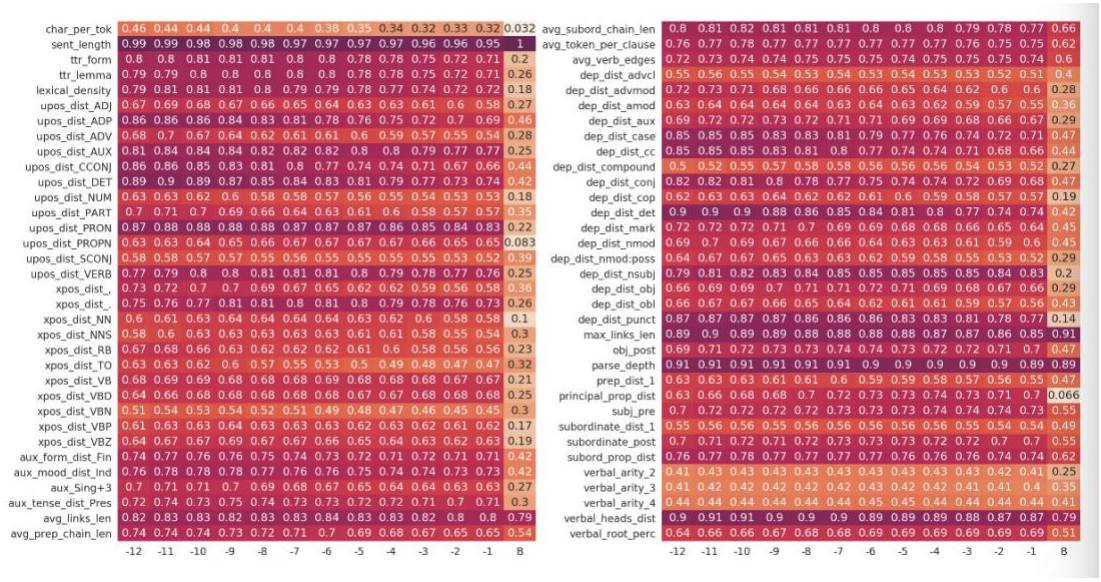


Figura 2.27. Punteggi di BERT-preaddestrato lungo i layer per le 68 feature linguistiche. La baseline è riportata nella colonna B

Entro l'ampio insieme di feature estratte dall'annotazione sintattica, si può notare che punteggi differenti sono riportati per le feature che si riferiscono, ad esempio, ai tipi di relazioni di dipendenza: quelli che collegano una POS funzionale alla sua testa (es. *dep_dist_case*, *dep_dist_cc*, *dep_dist_conj*, *dep_dist_det*) sono predette in maniera migliore rispetto alle altre relazioni, come *dep_dist_amod*, *advcl*. Inoltre, tra il gruppo *VerbPredicate*, sono ottenuti punteggi di Spearman più bassi per le feature che codificano informazioni su sottocategorie di predicati verbali, come ad esempio la distribuzione dei verbi per *arity* (*verbal_arity_2*, *3*, *4*), che rimane stabile lungo i layers.

Partendo dalla stessa versione preaddestrata del modello usata per questi esperimenti, è stato condotto un processo di *fine-tuning* per ciascuno dei 10 sottoinsiemi del corpus NLI preso in esame. Dopo aver constato le ottime performance ottenute sul task di *natural language identification* (con una media di accuratezza del 78.9% tra le diverse lingue), per ciascuna rappresentazione delle frasi del dataset UD (da parte dei modelli) dopo il processo di *fine-tuning*, sono stati nuovamente eseguiti *probing task*.
La figura 2.28 riporta la media dei valori di correlazione di ciascun layer per tutti i probing task ottenuti con BERT-base e gli altri modelli dopo il processo di *fine-tuning*.

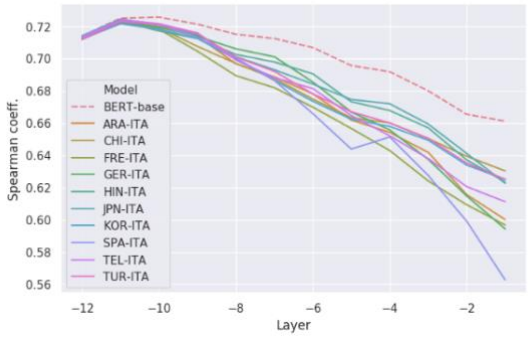


Figura 2.28. Media degli score per i modelli preaddestrato (BERT-base) e *fine-tuned*

Si può notare che le rappresentazioni imparate dopo il fine-tuning tendono a perdere precisione nel codificare l'insieme di feature scelto. Questo è particolarmente visibile nei livelli più alti e suggerisce che il modello memorizza informazioni specifiche per il task al costo di perdere la sua capacità di codificare conoscenza generale sul linguaggio. Per approfondire quale fenomeno è maggiormente coinvolto in questa perdita, è stata calcolata la differenza tra i risultati dei *probing task* ottenuti prima e dopo il processo di fine-tuning, concentrandosi in particolare sui punteggi ottenuti dalle rappresentazioni dell'output layer. Per ciascuna feature, la figura 2.29 riporta le differenze tra i valori ottenuti con il modello preaddestrato e fine-tuned.

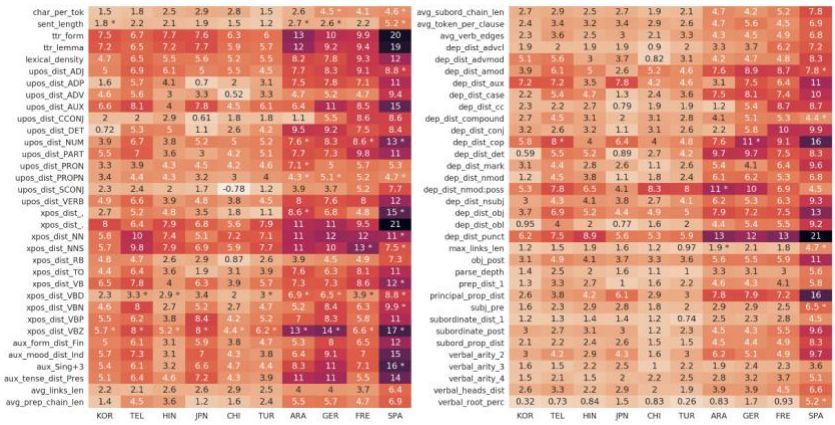


Figura 2.29. Differenze per il layer di output tra i punteggi di Spearman ottenuti tra BERT–base e i modelli fine-tuned (moltiplicate per 100). Le variazioni statisticamente significative (Wilcoxon Rank-sum test) sono marcate (*)

Si può notare che quando il task riguarda il distinguere l'italiano dal tedesco o il francese dallo spagnolo, BERT perde molta della sua conoscenza codificata in tutte le feature considerate. Questo è particolarmente evidente per le feature morfosintattiche

e le feature correlate alla varietà lessicale. Sembra che per i linguaggi tipologicamente simili BERT necessiti di conoscenza più specifica al task maggiormente codificata a livello di informazione morfosintattica piuttosto che a livello strutturale.

Contrariamente, la perdita è meno pronunciata e in molti casi non significativa per i modelli *fine-tuned* sulla classificazione di linguaggi più distanti (come coreano-italiano o turco-italiano). In questo caso, le performance sono abbastanza stabili e suggeriscono che queste feature sono ancora utili per il task considerato. Inoltre, è interessante notare che le classi di feature che decrescono significativamente in tutti i modelli sono quelle che codificano la conoscenza relativa al tempo verbale. è evidente il caso della terza persona singolare dei verbi al presente (*xpos_dist_VBZ*) e dei verbi al passato (*xpos_dist_VBD*). Una possibile spiegazione può essere correlata al genere testuale del corpus in esame.

Infine, per rispondere all'ultima domanda di ricerca (ossia se la conoscenza implicita del modello ha effetti sulla capacità di risolvere uno specifico task a valle) è stato diviso ciascun sottoinsieme del dataset NLI in due gruppi (le frasi correttamente classificate secondo la giusta L1 e quelle classificate in modo errato). Per i due gruppi di ciascun sottoinsieme, sono stati eseguiti *probing task* usando la versione preaddestrata di BERT-base e quella successiva al processo di *fine-tuning* su ciascuno specifico task di NLI. Per ciascuna frase dei due gruppi è stata calcolata la variazione tra le feature reali e quelle predette in modo da ottenere due liste di errori.

Usando infine il test *Wilcoxon Rank-sum* per verificare se le due liste avevano la stessa distribuzione, è risultato che molto più della metà delle feature variano in modo statisticamente significativo tra frasi classificate correttamente e non correttamente. Questo suggerisce che la competenza linguistica di BERT sui due gruppi di frasi è diversa.

Per approfondire l'analisi della differenza, è stata poi calcolata l'accuratezza ottenuta da BERT in termini di MSE (*Mean Square Error*) solo per l'insieme delle feature che variano in un modo significativo. La figura 2.30 riporta la percentuale delle feature per cui la MSE delle frasi correttamente classificate (*MSE Pos*) è minore di quelle non correttamente classificate (*MSE Neg*). Questo mostra che la capacità di BERT di

codificare diverso tipo di informazione linguistica può influenzare le sue predizioni: più BERT immagazzina informazione linguistica leggibile nelle rappresentazioni che crea, maggiore sarà la capacità di predire la corretta L1. In più, si può notare che questo è vero anche (e specialmente) usando il modello pre-trainato.

In altre parole, questo risultato suggerisce che la valutazione della conoscenza linguistica codificata in una versione pre-addestrata di BERT su una specifica sequenza in input può essere un indicatore perspicace della sua capacità di analisi rispetto a un compito a valle.

Figura 2.30. Percentuale delle feature di probing per cui la MSE delle frasi correttamente classificate da BERT-base (pre-train) e fine-tuned è minore delle frasi non corrette.

Per concludere, in risposta alle tre domande di ricerca iniziali si può dire che:

1) nella versione preaddestrata di BERT sono codificate un ampio insieme di fenomeni linguistici lungo i 12 layer, ma l'ordine in cui le *feature* di *probing* sono immagazzinate nelle sue rappresentazioni interne non riflette necessariamente la tradizionale divisione dei livelli di annotazione;
2) BERT tende a perdere precisione nel codificare questo set di *feature di probing* dopo essere stato *fine-tuned* sullo specifico task di *natural language identification*;
3) l'informazione della conoscenza implicita codificata da BERT ha effetti sulla sua capacità di risolvere task specifici.

3. Il compito dell'accettabilità/complessità linguistica

Nel presente studio sono stati presi in considerazione due compiti di analisi linguistica che riguardano il giudizio di *accettabilità* e *complessità* di una frase. Questa scelta è motivata dalla richiesta, sempre maggiore da parte di tutta la comunità scientifica, di risorse e sistemi in grado di predire le due valutazioni a livello di frase. Da un punto di vista dell'elaborazione naturale del linguaggio (NLP) l'interesse è aumentato grazie all'avanzare dei sistemi di generazione automatica (come sistemi di traduzione o semplificazione del testo) basati maggiormente su algoritmi di *deep neural network*. Infatti, metodi in grado di valutare la qualità delle frasi generate automaticamente o il loro grado di accettabilità e complessità si sono mostrati fondamentali per la valutazione di sistemi automatici. Da una prospettiva più teorico/linguistica, invece, dataset controllati contenenti giudizi di accettabilità e analizzati con tecniche di *machine learning* si sono rilevati utili per testare in che misura una deviazione sintattica e semantica può essere indotta ai soli dati del corpus (soprattutto per i fenomeni a bassa frequenza).

In questo scenario, la campagna di valutazione dell'elaborazione del linguaggio naturale, EVALITA, nel 2020 ha proposto un task per la stima dell'accettabilità e della complessità per l'italiano, *AcCompl-it* (Brunato, Chesi e altri, 2020). Più nello specifico, lo scopo era quello di sviluppare metodi di valutazione per classificare frasi italiane in accordo con l'accettabilità e la complessità, che possono essere viste come due semplici misure numeriche associate alle produzioni linguistiche. Insieme al task, è stato reso disponibile un insieme di frasi annotate con giudizi umani di accettabilità e complessità, che, ad oggi, risulta essere il primo corpora che combina insieme entrambe le valutazioni.

Segue una breve descrizione del task *AcCompl-it*, del dataset rilasciato per il suo svolgimento, dei risultati ottenuti nella campagna di valutazione e, infine, della descrizione dei dati utilizzati nel presente studio, in parte derivanti da quelli rilasciati per EVALITA.

3.1 Task AcCompl-it

AcCompl-it è un task di predizione dove è stato chiesto ai partecipanti di stimare il punteggio medio di accettabilità e complessità di un insieme di frasi precedentemente giudicate da parlanti su una scala Likert da 1 a 7 e, se possibile, predire l'esatto errore standard (SE) tra le annotazioni. Più in particolare, il task si è articolato in tre sotto-task:

1. Un task di predizione dell'*accettabilità*, dove i partecipanti devono stimare il punteggio di accettabilità delle frasi (insieme al loro errore standard). In questo caso, 1 corrisponde al più basso grado di accettabilità, mentre 7 corrisponde al livello più alto. L'assegnazione dello score su una scala graduale è in linea con la definizione dell'accettabilità percepita dai parlanti. Secondo la letteratura, infatti, l'accettabilità è un concetto vicino alla grammaticalità, ma mentre la grammaticalità corrisponde alla buona formazione sintattica della frase da un punto di vista teorico e, tipicamente è intrepretata come proprietà binaria, l'accettabilità può dipendere da diversi fattori, come la sintassi, la semantica, la pragmatica e altri aspetti non linguistici;
2. Un task di predizione della *complessità*, dove i partecipanti devono stimare il punteggio di *complessità* delle frasi (insieme al loro errore standard). In questo caso 1 corrisponde al minor livello di complessità, mentre 7 indica il grado maggiore. Similmente al task della predizione dell'accettabilità, l'uso della scala Likert come strumento per raccogliere valori è motivato dall'assunzione che la complessità di una frase è un gradiente, piuttosto che un concetto binario;
3. Un task "aperto" dove i partecipanti devono modellare fenomeni linguistici correlati con il voto umano sull'accettabilità e complessità delle frasi del dataset fornito;

Questi tre sottotask sono indipendenti, anche se è stata incoraggiata la partecipazione a ciascuno di questi.

3.1.1 Dataset AcCompl-it

Come già accennato precedentemente, insieme al task è stato rilasciato un insieme di frasi annotate con giudizi di *accettabilità* e di *complessità*. Più in particolare è stato reso disponibile un *dataset per l'accettabilità* e uno per la *complessità* (contenente anche 672 frasi generate automaticamente comuni al dataset dell'accettabilità).

Nello specifico il *dataset per l'accettabilità* contiene 1683 frasi italiane annotate con giudizi umani su una scala Likert da 1 a 7. Il numero di annotazioni per ciascuna frase va da 10 a 85 con una media di 16.38. Questo dataset è costruito unendo i dati di quattro studi psicolinguistici (basati su variazioni minime di opposizioni linguistiche controllate su diversi livelli di grammaticalità) con un insieme di 672 frasi generate da template.
Un primo insieme di questo dataset (128 frasi), preso dallo studio di Chesi e Canal del 2019, si focalizza su opposizioni della persona nell'oggetto delle dislocazioni delle dipendenze (1):

(1) {Sono | Siete} {gli | voi} architetti che {gli | voi} ingegneri {hanno | avete} consultato.

Più in particolare, la costruzione (1) va a testare la percezione del livello di accettabilità di frasi in cui il soggetto della frase (relativa) incassata può essere in accordo o meno (rispetto al tratto di persona) con il soggetto della frase principale; i due soggetti possono essere introdotti dallo stesso elemento (pronome o articolo) o da due elementi diversi (uno dal pronome e l'altro dall'articolo, o viceversa). In questo caso nessuna delle due varianti crea una frase agrammaticale ma ciascuna variante dovrebbe essere percepita con un livello diverso di accettabilità.

Un secondo insieme di frasi (515) è ricavato dagli studi presentati in (Greco e altri, 2020), coinvolgendo costruzioni copulative (es: canoniche (2a) vs. inverse (2b)) (Moro, 1997):

(2) a. Le foto del muro sono la causa della rivolta.
    b. La causa della rivolta sono le foto del muro.

Questo sottoinsieme contiene anche frasi dichiarative e interrogative (sì/no) con una struttura verbale minima (mettendo in opposizione soggetti preverbali e postverbali in predicati inergativi (3a), inaccusativi (3b) e transitivi (3c)):

(3) a. I cani hanno abbaiato | Hanno abbaiato i cani.

b. Gli autobus sono partiti | Sono partiti gli autobus.

c. Le bambine hanno mangiato il dolce | Hanno mangiato le bambine il dolce.

Il terzo insieme di frasi (320) è basato sullo studio dell'accordo/disaccordo tra persona e numero del soggetto e del verbo (Mancini e altri, 2018):

(4) Qualcuno ha detto che io {scrivo | scriviamo} una lettera.

Il quarto gruppo di frasi (48) contiene diverse violazioni nelle proposizioni interrogative:

(5) {Cosa | Quale edificio} ti chiedi {chi | quale ingegnere} abbia costruito?

L'ultimo insieme di 672 frasi è stato generato creando tutte le possibili combinazioni di parole da template strutturali progettati per testare l'accettabilità di differenti costruzioni. Ad esempio, dalla costruzione (6) alla (9) una delle due varianti proposte crea una frase agrammaticale, quindi con basso livello di accettabilità. Tutte queste costruzioni fanno riferimento alla teoria del movimento, maturata nel quadro della grammatica generativa, che definisce la possibilità di dislocare elementi nell'ambito della struttura della frase (Graffi e altri, 2003; Rizzi e altri, 1979).

Nello specifico, le costruzioni 6 e 7 si riferiscono allo stesso macro-fenomeno, cioè la frase diventa agrammaticale quando la posizione (*gap*) lasciata dall'elemento dislocato è riempita.

In particolare, la 6.a è una domanda nella quale il complemento oggetto del verbo principale ('descrivere') è già espresso come elemento interrogativo ('quale problema' o 'cosa'). Di conseguenza, quando è presente anche il pronome ('lo'), sempre in funzione di complemento oggetto, si crea una agrammaticalità perché ci sono due elementi che realizzano in maniera esplicita la stessa funzione.

La 6.b è analoga alla 6.a, ma non in forma di domanda. L'elemento dislocato ('questo problema') a sinistra ha il ruolo di topic.

Nella 7, l'elemento dislocato ('chi') è il complemento oggetto di entrambi i verbi ('chiamare' e 'medicare') tra loro coordinati. Di conseguenza, l'agrammaticalità è data dalla presenza, in una o entrambe le frasi congiunte, di un pronome esplicito ('lo') che svolge la stessa funzione di complemento oggetto.

(6)

a. {Cosa | Quale problema} lo studente dovrebbe descriver(e) {_| -lo | questo problema}?

b. Questo problema lo studente dovrebbe descriver(e) {_, -lo | questo problema}.

(7)

a. Chi … Maria vuole chiamar(e) {_ | -lo} e il dottore mendicar(e) {_ | -lo}?

Nella costruzione 8, l'agrammaticalità è data dallo spostamento a sinistra dell'elemento dislocato ('quale provvedimento'), che costituisce il complemento oggetto del verbo 'prendere', oltre la frase 'Maria ha saputo', chiamata isola, che è una *embedded Wh-clauses*, cioè una frase incassata. In presenza di isole, sempre secondo la teoria linguistica del movimento, si attivano delle restrizioni allo spostamento, che se violate creano agrammaticalità.

(8)

a. Quale provvedimento Maria ha saputo {che | dove | perché | quando} il ministro prenderà?

Nella 9, le varianti dipendono dal fatto che l'elemento dislocato ('il compagno di classe') svolge la funzione di complemento oggetto o di soggetto della frase relativa. In questo caso, nessuna delle due varianti è completamente agrammaticale, ma quella in cui l'elemento dislocato svolge la funzione di complemento oggetto dovrebbe essere percepita come meno accettabile.

(9)

a. Carlo conosceva bene il compagno di classe che {incontrare _ divertiva sempre Anna | Anna voleva sempre incontrare _}.

Infine, nella 10, l'agrammaticalità è data da pronomi come *nessuno* o *alcunché* posti prima di una domanda con negazione o avverbi come *mai* posti successivamente in una frase semplice o subordinata (preceduta da una negazione):

(10)

a. {Maria | Nessuno} si aspetta che qualcuno possa aver {già | mai} finito questo esercizio(?)

Il *dataset per la complessità*, invece, comprende 2530 frasi italiane annotate su una scala Likert da 1 a 7 sulla percezione della complessità. Il numero di annotazioni per

frase va da 11 a 20 con una media di 16.75. Il corpus è stato diviso internamente in due sottoinsiemi rappresentativi di due diverse tipologie di dati: 1858 frasi estratte da corpora, 672 frasi generate artificialmente estratte dal dataset dell'*accettabilità* e scelte per coprire lo spettro di fenomeni linguistici rappresentato nei template.
Il primo gruppo di frasi deriva in larga misura (1128 frasi su 1858) dalla sezione giornalistica della Italian Stanford Dependency Treebank (ISDT) (Bosco e altri, 2013) annotata con giudizi di complessità da Brunato e altri (2018). Le restanti frasi si suddividono in frasi rappresentative di una varietà linguistiche non standard (estratte da PoSTWITA e TWITTIRÒ (Sanguinetti e altri, 2018; Cignarella e altri, 2019)), frasi interrogative estratte dalla sezione "quest" della treebank ISDT, discorsi diretti che includono trascrizioni dei dibattiti del parlamento europeo (estratti dalla sezione "europarl" di ISDT) ed estratti di testi letterari (contenuti nella UD Italian VIT (Delmonte e altri, 2007)).
La scelta di annotare una porzione di dati sia con punteggi di accettabilità che di complessità è stata esplicitamente motivata dal tentativo empirico di investigare se c'è una relazione tra le due proprietà e se la complessità è giudicata differentemente nel caso di costruzioni malformate.

Per la raccolta delle annotazioni sui giudizi di accettabilità e complessità da parte dei parlanti madrelingua italiani, è stato fatto affidamento a tecniche di *crowdsourcing* usando diverse piattaforme. Più nello specifico, per il dataset dell'*accettabilità*, l'insieme delle frasi estratto da studi psicolinguistici è stato annotato usando una piattaforma online basata su script jsPsych (De Leeuw, 2015). Per il dataset della *complessità*, invece, alcune annotazioni sono state eseguite sulla piattaforma CrowdFlower, mentre altre utilizzando Prolific[4]. Per rendere le annotazioni comparabili, l'intero processo è stato suddiviso in diversi task, ciascuno consistente nell'annotazione di circa 200 frasi mescolate randomicamente tra le differenti tipologie. Per tutti i compiti, è stato chiesto ai partecipanti dell'annotazione di leggere ciascuna frase e rispondere alle seguenti domande:

1. "Quanto è complessa questa frase da 1 (semplicissima) a 7 (molto difficile)?"

[4] www.prolific.co

2. “Quanto è accettabile questa frase da 1 (completamente agrammaticale) a 7 (perfettamente grammaticale?”

Dopo aver raccolto tutte le risposte, sono state escluse quelle assegnate in meno di 10 minuti, in quanto considerate poco accurate.

La tabella 3.1 mostra il valore medio, minimo, massimo e la deviazione standard per le etichette di complessità e accettabilità dell’intero dataset.

	COMPL		**ACCEPT**	
	SCORE	SE	SCORE	SE
μ	3.12	0.332	4.45	0.36
σ	1.04	0.08	1.7	0.14
min	1	0	1.13	0
max	6.46	0.63	7	0.74

Tabella 3.1. Statistiche raccolte per i due corpora del dataset AcCompl-it

Come si può notare, i valori della complessità sono mediamente inferiori rispetto a quelli dell’accettabilità. Inoltre, per la complessità il valore minimo della scala Likert (1) – che dovrebbe essere usato per etichettare frasi percepite come molto semplici – è stato assegnato solamente due volte, specificatamente per le frasi:

(11) “Dimmi il nome di una città finlandese.”

(12) “Quali uve si usano per produrre il vino?”

Per quanto riguarda il corpus dell’accettabilità, invece, il valore più alto (7) – assegnato alle frasi completamente accettabili – è stato attribuito 26 volte. Di seguito si riportano due esempi chiarificatori:

(13) “Le sorelle sono sopravvissute.”

(14) “I lupi hanno ululato.”

Le frasi che, invece, sono state giudicate come più complesse (con un punteggio di 6.46) e meno accettabili (con un punteggio di 1.55) in ciascun dataset sono le seguenti:

(15) “Chi è che lui ha affermato che il professore aveva detto che lo studente avrebbe dovuto considerare questo candidato?”

(16) “Il falegname è arrivato mentre noi montavo la mensola.”

Analizzando più a fondo i vari punteggi, si nota come per la complessità, gli score medi sono superiori per le frasi create per mostrare specifici pattern di accettabilità, dimostrando così che l'accettabilità influisce sulla percezione della complessità.
Tra le frasi delle treebank quelle estratte dai testi giornalistici (ISDT_news) sono giudicate in media come più complesse, mentre le frasi interrogative sono considerate come più semplici. Le frasi estrapolate da Twitter e quelle contenenti discorsi diretti, invece, hanno punteggi medi molto vicini tra di loro. Questo è in linea con l'analisi stilistica e linguistica, mostrando che il linguaggio dei social media eredita molte caratteristiche del linguaggio parlato.
Inoltre, per l'intero dataset dell'accettabilità, il coefficiente di correlazione di Spearman tra il punteggio di grammaticalità e di accettabilità è molto alto (0.83, $p<0.001$). Invece, il punteggio di complessità (delle 672 frasi valutate anche per accettabilità) ha una correlazione inferiore, ma comunque significativa con il punteggio di grammaticalità (0.34, $p<0.001$).

Per l'esecuzione di entrambi i task, la metrica di valutazione si è basata sul coefficiente di correlazione del rank di Spearman tra gli score dei partecipanti e il test set. In questo modo, sia per l'accettabilità che per la complessità, sono stati prodotti due diversi ranghi in accordo alle predizioni dei relativi score e all'errore standard. Il calcolo della baseline, invece, è stato definito in diverso modo:

1. Nel task dell'*accettabilità*, è stato scelto lo score associato da un regressore lineare SVM che usa come feature unigrammi o bigrammi di parole;
2. Nel task della *complessità*, invece, si è preferito lo score assegnato da un regressore lineare SVM che usa la lunghezza della frase come unica feature;

3.1.2 Risultati AcCompl-it

Il task AcCompl-it ha ricevuto tre sottomissioni per ciascun sotto task da parte di due diversi partecipanti. Purtroppo, nessuno ha partecipato al task "aperto". I sistemi dei due partecipanti hanno seguito approcci molto differenti: uno si è basato sul deep learning ed è stato addestrato su testi *raw* (Sarti, 2020), l'altro ha seguito regole (euristiche) applicate a feature semantiche e sintattiche estratte automaticamente dalle frasi (Delmonte, 2020). Nonostante la loro natura molto diversa, i due approcci presentano anche degli aspetti comuni, come l'affidamento a risorse esterne. Più nello

specifico, entrambi utilizzano frasi aggiuntive estratte da treebank italiane esistenti, sia per arricchire il training set originale con altri esempi annotati (come nel caso di Sarti) sia per verificare la frequenza di una data costruzione per poi utilizzare questa informazione tra le feature del sistema proposto (ItVenses).
Il sistema di Sarti ha ottenuto le migliori performance in entrambi i task usando un approccio simile al *multi-task learning*, cioè facendo leva sulle predizioni di un modello neurale del linguaggio per l'italiano (UmBERTO[5]), *fine-tuned* sui due task a valle, per ampliare il *development set* originale con un insieme maggiore di esempi non etichettati (estratti dalle treebank italiane disponibili). Il dataset esteso è stato poi suddiviso in diverse porzioni per addestrare un insieme di classificatori. Il risultate modello di *multi-task learning*, infine, è stato utilizzato per predire le etichette di complessità/accettabilità dell'originale test set.
Il sistema *ItVenses* di Delmonte, invece, parsa le frasi per ottenere una sequenza di costituenti e un insieme di feature semantiche a livello di frase (come la presenza dell'accordo, marcatori di negazione, fattività ecc.). Queste caratteristiche, insieme ai costituenti e alla loro frequenza nel training set e nella *Venice Italian Treebank*, vengono pesate con diverse euristiche e usate per fare predizioni. Le performance di ItVenses sono considerevolmente minori rispetto a Sarti (ottengono circa il 50% in meno nei dati psicolinguistici). Tuttavia, nel sottoinsieme dei dati artificiali, che ha frasi complesse ma molto meno diversificate nella struttura, il gap con il sistema vincente è minore (circa il 20% in meno).

Le buone performance del sistema vincente in entrambi i task non sono totalmente inaspettate, considerando soprattutto i risultati soddisfacenti ottenuti dalle reti neurali correnti in molteplici compiti di NLP. A questo proposito, vale la pena menzionare che, nel suo report, Sarti ha comparato le performance del migliore sistema basato sul *multi-task leaning* a quelle ottenute con una versione più semplice di UmBERTo con un *fine-tuning* sui due compiti di analisi linguistica, ottenendo già risultati molto buoni (il 90% e l'85% per le predizioni rispettivamente di accettabilità e complessità sul training set).

[5] https://github.com/musixmatchresearch/umberto

3.2 Dataset selezionati per questo studio

Per il presente studio sono stati messi a punto diversi modelli neurali al fine di determinare come varia la loro competenza linguistica dopo il processo di *fine-tuning*. Rispetto agli studi precedenti, come accennato precedentemente, i task considerati hanno riguardato compiti di valutazione dell'*accettabilità* e *complessità* di una frase. Più in particolare tutti gli esperimenti sono stati condotti su due dataset:

1. Uno utilizzato per i due compiti di analisi linguistica;
2. Sezioni dell'Italian Dependency Treebank (IUDT) per testare l'informazione linguistica precedente e successiva al processo di *fine-tuning*;

3.2.1 Dati per la valutazione di accettabilità/complessità

Per i task di valutazione dell'*accettabilità*/*complessità* sono state prese in considerazione le 672 frasi, generate artificialmente secondo i template descritti nella sezione precedente (3.1.1), comuni ai due dataset rilasciati per il task *AcCompl-it* di EVALITA 2020. Ad oggi, questo risulta essere il primo corpora per l'italiano che combina insieme entrambe le valutazioni.

La tabella 3.2 riassume i giudizi di *accettabilità* e complessità presenti nell'insieme delle frasi considerate:

	COMPL	**ACCEPT**
media	3.63	4.00
deviazione standard	0.95	1.49
min	1.42	1.38
max	6.46	7.00

Tabella 3.2. Statistiche dataset utilizzato per la valutazione dell'*accettabilità*/*complessità*

Come si può notare, anche per questo insieme di frasi, il punteggio medio della *complessità* è inferiore rispetto a quello dell'*accettabilità*, ma si ricorda che un punteggio più alto nel caso dell'accettabilità indica frasi più grammaticali, mentre un punteggio più basso nel caso della complessità è un fattore di maggior semplicità.

Inoltre, persino in questa porzione di dataset la deviazione standard risulta inferiore nel caso della *complessità* piuttosto che per l'*accettabilità*.

A scopo esemplificativo, si riportano la frase giudicata con un punteggio minimo di complessità (a), quindi più semplice, e accettabilità, quindi più agrammaticale (b):

a. "Carla dovrebbe menzionarlo";
b. "Chi è che Maria ha saputo perché Carlo incontrerà?";

Tra le frasi con punteggi massimi, invece, troviamo (c) valutata come frase più complessa e (d) come frase più accettabile:

c. "Chi è che lui ha affermato che il professore aveva detto che lo studente avrebbe dovuto considerare questo candidato?";
d. "Maria ti ha detto dove Carlo farà un sopralluogo";

Avendo a disposizione entrambi i punteggi per ciascuna frase presente nel dataset, si è calcolata la correlazione tra le due metriche, ottenendo uno score di -0.49. Questo risultato di correlazione suggerisce che più una frase è percepita come complessa, meno è accettabile.

Per indagare statisticamente quali fenomeni linguistici sono maggiormente correlati con i fenomeni di accettabilità e complessità, per ciascuna frase del dataset è stato calcolato, utilizzando la piattaforma *Profiling Ud*[6] (Brunato, 2020), un vettore di caratteristiche linguistiche[7]. Più nello specifico, questo tool ha permesso l'estrazione di molteplici feature che variano lungo i diversi livelli di descrizione linguistica, come ad esempio le proprietà *raw* del testo, le informazioni sulla varietà lessicale, la morfosintassi, la struttura verbale del predicato, le relazioni sintattiche e l'uso della subordinazione, per un totale di 92 caratteristiche. Sono state eliminate alcune feature non interessanti per il calcolo della correlazione di *Spearman* come il *nome del file*, il *numero di frasi* (dato che il valore corrispondente era sempre 1) o il *numero di tokens per frase* (dato che risultava essere un'informazione ridondante, visto che ogni periodo

[6] http://linguistic-profiling.italianlp.it/

[7] In appendice (capitolo 9) è presenta la lista di tutte le feature linguistiche estratte dalla piattaforma *Profiling UD* con la loro relativa spiegazione.

era costituito da una sola frase e quindi tale valore poteva essere ricavato semplicemente dal *numero di tokens*). Una volta ottenuti i vettori, è stata presa in esame ciascuna dimensione di questi per calcolare la correlazione di Spearman con i punteggi di accettabilità e complessità. Considerando solamente i punteggi con pvalue minore di 0.05, le tabelle 3.3 e 3.4 riportano le feature maggiormente correlate (sia in modo positivo che negativo) rispettivamente con *accettabilità* e *complessità*. Delle 92 feature iniziali, solamente 61 hanno ottenuto un punteggio significativo per l'accettabilità e 60 per la complessità.

Feature	**Correlazione di Spearman**	**P-value**
verbs_num_pers_dist_Sing+3	0.31987206676080800	1.88056561363873E-17
upos_dist_ADP	0.3182200118057880	2.80043228106071E-17
dep_dist_case	0.3008733217519950	1.58137658658332E-15
dep_dist_advmod	0.2937109289627200	7.7416906111968E-15
upos_dist_ADV	0.2937109289627200	7.7416906111968E-15
…	…	…
n_tokens	-0.2792121163797960	1.68552861981322E-13
dep_dist_acl	-0.2979633148038260	3.03127530348349E-15
verbal_head_per_sent	-0.3262903618299540	3.91004670945485E-18
avg_max_depth	-0.3509832528657330	6.49285764578093E-21

Tabella 3.3. Feature maggiormente correlate con il punteggio di *accettabilità*

Feature	**Correlazione di Spearman**	**P-value**
n_tokens	0.6057219227999740	1.58053111339122E-68
aux_form_dist_Part	0.41344027842416700	3.9518212821622E-29
aux_tense_dist_Past	0.41344027842416700	3.9518212821622E-29
subordinate_proposition_dist	0.4107145260381220	9.84701526675902E-29
verbs_tense_dist_Past	0.37849145005405800	2.58254799162631E-24
…	…	…
lexical_density	-0.2946350584802920	6.32292212283778E-15
principal_proposition_dist	-0.4029655676778470	1.26107657148179E-27
dep_dist_punct	-0.4172297123535660	1.09512601407716E-29
upos_dist_PUNCT	-0.4172297123535660	1.09512601407716E-29

Tabella 3.4. Feature maggiormente correlate con il punteggio di *complessità*

È interessante notare come tra le caratteristiche maggiormente correlate in maniera positiva con il punteggio di *accettabilità* troviamo *verb_num_pers_dist_Sing+3*, ossia la distribuzione dei verbi in accordo al loro numero e persona. Questo significa che i verbi alla terza persona singolare sono percepiti come maggiormente accettabili. Per questa metrica risultano importanti anche *upos_dist_ADP* relativo alla distribuzione delle preposizioni, *dep_dist_case* relativo alla distribuzione delle relazioni tra i marcatori e il nome da cui dipendono o che introducono, *dep_dist_advmod* relativo alla distribuzione dei modificatori avverbiali e *upos_dist_ADV* relativo alla distribuzione degli avverbi. Tra le caratteristiche linguistiche che invece rendono una frase meno accettabile troviamo il *numero di tokens* (più una frase è lunga e meno è accettabile), il numero elevato di teste verbali per frase e la lunghezza elevata dell'albero delle dipendenze.

Per quanto riguarda la complessità, invece, le feature maggiormente correlate in senso positivo risultano il *numero di tokens* (in questo caso più una frase è lunga, più è complessa), la distribuzione degli *ausiliari al participio*, al *passato* o, più in generale, dei verbi al passato (*verbs_tense_dist_Past*) e la distribuzione delle frasi subordinate. Tra le caratteristiche che rendono invece una frase meno complessa troviamo l'alta *densità lessicale* (cioè il rapporto tra le parole piene e il totale delle parole del documento), la *distribuzione delle proposizioni principali* e feature relative alla punteggiatura (*dep_dist_punct*, *upos_dist_PUNCT*).

Confrontando, i ranking delle feature maggiormente correlate con i due task, tramite una correlazione per ranghi di Spearman, si è ottenuto un punteggio di -0.37. Questo valore negativo suggerisce, ancora una volta, che le feature che rendono la percezione della frase come più complessa, al contempo contribuiscono a renderla meno accettabile.

Nel quarto capitolo si descriverà quanto sono in grado di predire punteggi di accettabilità e complessità sia un regressore lineare che un modello neurale del linguaggio.

3.2.2 Italian Dependency Treebank (IUDT)

Per testare l'informazione linguistica precedente e successiva al processo di *fine-tuning* sono state utilizzate cinque sezioni dell'Italian Universal Dependency Treebank (IUDT), che riuniscono vari testi. Più nello specifico, si è ottenuta una rappresentazione di ciascuna frase della treebank con i diversi modelli precedentemente messi a punto sui task di *accettabilità*/*complessità* e successivamente, per ciascuna rappresentazione sono stati eseguiti 82 *probing task*, utilizzando un modello di regressione lineare (*LinearSVR*) come spiegato più nel dettaglio nel capitolo 5.2.

Il dataset preso in esame comprende un totale di 35480 frasi rappresentative di diversi generi e varietà testuali. La tabella 3.5 illustra la composizione del dataset più nel dettaglio.

Nome sezione	**Tipo di testo**
ParTUT	Multi-genere
VIT	Multi-genere
ISDT	Multi-genere
ISDT_tanl	Giornalistico
ISDT_tut	Legale/Giornalistico/Wiki
ISDT_quest	Frasi interrogative
ISDT_2parole	Notizie italiane semplificate
ISDT_europarl	Atti del parlamento europeo
PoSTWITA	Tweet
TWITTIRÒ	Tweet ironici
Totale frasi:	35480

Tabella 3.5. Sezioni della Italian Universal Dependency Treebank (IUDT)

Le cinque sezioni della treebank (IUDT) considerate consistono in: ParTUT (Sanguinetti e Bosco, 2015), VIT (Delmonte e altri, 2007), ISDT (Bosco e altri, 2013), PoSTWITA (Sanguinetti e altri, 2018) e TWITTIRÒ (Cignarella e altri, 2019).

Nella tabella 3.5, inoltre, si distinguono diversi sotto-corpora di ISDT in accordo alle specifiche varietà del linguaggio che rappresentano, come le trascrizioni del parlato spontaneo (*ISDT_europarl*), le domande (*ISDT_quest*) o il linguaggio semplificato (*ISDT_2parole*).

4. Predizione dell'accettabilità e della complessità

In questo capitolo si illustrerà quanto sia un regressore lineare che un modello neurale del linguaggio (BERT) siano in grado di eseguire i due compiti di analisi linguistica riguardanti la valutazione dell'accettabilità e della complessità delle frasi del dataset descritto nel capitolo 3.2.1.

Più nello specifico, inizialmente, è stata valutata la capacità di un regressore lineare (un Support Vector Regression con kernel *linear*) nel predire i punteggi su una scala Likert da 1 a 7 di accettabilità e complessità, assegnati alle 672 frasi del dataset rilasciato per *AcCompl-it* (EVALITA 2020). Dopodiché, si è cercato di capire quanto la presenza di feature linguistiche esplicite (ricavate da *Profiling UD* per ciascuna frase del dataset) influenzasse le performance del modello di regressione. Infatti, le performance del SVR sono state successivamente confrontate con quelle di un modello neurale del linguaggio che non utilizza feature linguistiche esplicite.

4.1 Analisi con un regressore lineare

Al fine di valutare le prestazioni di un regressore lineare SVR (Support Vector Regression), per ciascuna delle 672 frasi del dataset, sono state estratte le stesse 92 feature utilizzate per il calcolo di correlazione con il punteggio di accettabilità/complessità illustrate nel capitolo 3.2.1. Per l'esecuzione di questo processo è stata utilizzata ancora una volta la piattaforma *Profiling UD*[8] e sono state nuovamente scartate, dalle caratteristiche linguistiche totali calcolate dal sistema, quelle ritenute meno significative (come *n_sentences*, *subordinate_pre*, *dep_dist_root*, *tokens_per_sent*).

Al termine di questo procedimento, è stato ricavato un vettore di feature linguistiche per ciascuna frase. Più nello specifico, si sono ottenuti 672 vettori di 92 dimensioni.

[8] In appendice (cap. 9) è presente una legenda completa delle feature linguistiche estratte con *Profiling UD*.

Prima di procedere con l'addestramento e il test del regressore lineare, i valori di ciascuna caratteristica linguistica sono stati normalizzati con il *MinMaxScaler* di *scikitLearn*[9]. Questo metodo trasforma ciascuna feature individualmente in modo da ottenere valori compresi tra zero e uno, applicando tale formula matematica:

$$X_{scaled} = X_{std} \times (\max - \min) + min$$

dove X corrisponde all'insieme dei dati di training, *min* e *max* rispettivamente al valore minimo e massimo di ciascuna feature e, infine, X_{std} è:

$$X_{std} = \frac{X - X.\min(axis = 0)}{X.\max(axis = 0) - X.\min(axis = 0)}$$

A questo punto, il dataset è stato suddiviso in 12 k-fold per effettuare una *k-fold cross-validation*, ossia un metodo statistico per valutare il modello di machine learning effettuando una procedura di ricampionamento. In particolare, questo processo prevede la suddivisione in *k-gruppi* di uguale numerosità e, a ripetizione, un gruppo è utilizzato come test-set per il modello, mentre la restante parte come insieme di addestramento. In questo modo, si allena il modello per ognuna delle *k* parti evitando problemi di *overfitting* o di campionamento asimmetrico (tipico della suddivisione dei dati in due sole parti).

Nel nostro caso, a rotazione $\frac{1}{12}$ del dataset è stato utilizzato come test (ossia 56 vettori di feature) e i restanti $\frac{11}{12}$ come training (616 vettori di feature). Per ciascun *k-fold* è stato fatto il *fitting* del modello di regressione (SVR) sui dati di training, dove sono stati passati 616 vettori di feature come dati (*x_train*) e come valori di predizione corrispondenti i punteggi rispettivi di accettabilità o complessità (*y_train*). Successivamente al *fitting* è stata eseguita la predizione sui valori del test e, quindi, passando al regressore l'insieme delle 92 feature linguistiche per le 56 frasi di test, si sono predetti 56 valori di accettabilità o complessità per ciascun *k-fold*. Una volta completato il processo per ciascun *k-fold*, per stimare la capacità del modello nel predire i valori di accettabilità o complessità date le 92 feature linguistiche per ciascuna frase, è stata fatta una correlazione di Spearman tra i valori predetti e quelli reali. La tabella 4.1 illustra i risultati ottenuti. Come si può notare, un regressore

[9] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

lineare è molto più capace nel predire punteggi di complessità (0.70), piuttosto che accettabilità (0.59), dato che i valori predetti sono molto più correlati a quelli reali.

	ACCETTABILITÀ	**COMPLESSITÀ**
SCORE	0.59	0.70

Tabella 4.1. Performance SVR nel predire i punteggi di accettabilità e complessità

L'intero procedimento appena descritto è stato eseguito due volte, ossia una volta per stimare le performance di valutazione dell'accettabilità e una volta per la complessità. Come regressore è stato utilizzato SVR di *ScikitLearn*[10], come metodo di ricampionamento *KFold* sempre di *ScikitLearn*[11] e come libreria per il calcolo della correlazione *corr* di *Pandas*[12].

Successivamente, per comprendere meglio quali feature linguistiche hanno guidato maggiormente il modello di regressione lineare nell'assegnazione del punteggio di accettabilità/complessità, è stato addestrato l'SVR sull'intero insieme di dati (tutti i 672 vettori) ed è stato estratto il parametro *coef*, che indica i pesi assegnati a ciascuna feature. Le tabelle 4.2 e 4.3 indicano le 10 feature con punteggio maggiore rispettivamente per l'accettabilità e per la complessità.

Feature	**Peso**
dep_dist_aux	2.0130535341354400
verbs_tense_dist_Fut	1.8442204648394000
upos_dist_AUX	1.2893686518145600
verb_edges_dist_5	1.2832673781934100
verbs_num_pers_dist_Sing+3	1.2810255541290100
verbs_mood_dist_Ind	1.057503948646660
avg_verb_edges	1.009600769701170
dep_dist_mark	0.9495528464261940
upos_dist_ADP	0.9031385355596120
verbs_form_dist_Part	0.8689284591982260

Tabella 4.2. Feature dell'SVR con peso maggiore per l'accettabilità

[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[11] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

[12] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

Feature	Peso
n_tokens	2.1130448731922800
verbal_head_per_sent	1.3714521673387100
char_per_tok	1.1820384146056400
avg_token_per_clause	1.1077469114340200
dep_dist_nmod	1.0586434940913100
upos_dist_PRON	0.6602749012275440
verbs_tense_dist_Past	0.6311193261037940
subordinate_proposition_dist	0.5025475642815470
aux_form_dist_Inf	0.4984256998572360
upos_dist_SCONJ	0.4779254095249630

Tabella 4.3. Feature dell'SVR con peso maggiore per la complessità

Come si può notare dalla tabella 4.2, la feature che ha maggior peso nella predizione del valore di accettabilità è *dep_dist_aux*, cioè la distribuzione media della relazione sintattica tra ausiliare e predicato. Risultano comunque rilevanti anche *verbs_tense_dist_Fut* (relativa alla distribuzione dei verbi al tempo futuro), *upos_dist_AUX* (la distribuzione degli ausiliari), *verb_edges_dist_5* (la distribuzione dei verbi con valenza 5), *verbs_num_pers_dist_Sing+3* (la distribuzione dei verbi alla terza persona singolare), *verbs_mood_dist_Ind* (la distribuzione dei verbi al modo indicativo), *avg_verb_edges* (la valenza verbale), *dep_dist_mark* (la distribuzione dei marcatori), *upos_dist_ADP* (la distribuzione delle preposizioni), *verbs_form_dist_Part* (la distribuzione dei verbi al participio). È interessante osservare che le feature *verbs_num_pers_dist_Sing+3* e *upos_dist_ADP* erano già risultate tra le più correlate per la valutazione dell'accettabilità, entrambe con un punteggio di 0.32. Considerando che tra le feature più importanti per l'SVR, le prime sette sono caratteristiche verbali e tra le caratteristiche linguistiche più statisticamente correlate con la valutazione dell'accettabilità vi è, al primo posto, la distribuzione dei verbi alla terza persona singolare, si può dedurre che per la valutazione dell'accettabilità giocano un ruolo fondamentale i verbi.

Per quanto riguarda la complessità, invece, la feature con peso maggiore risulta essere *n_tokens* (ossia il numero di tokens di una frase). È rilevante considerare che questa feature si era mostrata la più correlata con il compito della complessità, con un

punteggio di 0.60. Sempre nella tabella 4.3, le altre caratteristiche linguistiche che si sono rivelate importanti per la valutazione della complessità sono *verbal_head_per_sent* (la distribuzione media delle teste verbali nel documento), *char_per_tok* (il numero di caratteri per token), *avg_token_per_clause* (lunghezza media delle clausole, calcolata in termini di numero medio di token per clausola), *dep_dist_nmod* (la distribuzione delle relazioni sintattiche tra i nomi e i modificatori nominali), *upos_dist_PRON* (la distribuzione dei pronomi), *verbs_tense_dist_Past* (la distribuzione dei verbi al passato), *subordinate_proposition_dist* (la distribuzione delle proposizioni subordinate), *aux_form_dist_Inf* (la distribuzione dei verbi ausiliari all'infinito), *upos_dist_SCONJ* (la distribuzione delle congiunzioni subordinati). Oltre a *n_tokens*, che come descritto precedentemente era già presente tra le feature maggiormente correlate con la complessità, tra queste appena elencate troviamo con un punteggio elevato di correlazione anche *verbs_tense_dist_Past* e *subordinate_proposition_dist* (rispettivamente con 0.38 e 0.41).

Per valutare quanto sono simili i ranking delle feature maggiormente correlate con i due compiti e delle feature con maggior peso secondo il modello di regressione, è stata calcolata la correlazione per ranghi di Spearman, considerando indipendentemente i due task. Più in particolare, è stata prima misurata la correlazione tra le feature pesate (del regressore per l'accettabilità) e quelle maggiormente correlate al compito di accettabilità e, successivamente, è stata eseguita la stessa operazione per la complessità. La tabella 4.4 illustra i punteggi di correlazione ottenuti.

	ACCETTABILITÀ	**COMPLESSITÀ**
SCORE	0.34	0.30

Tabella 4.4. Correlazione per ranghi di Spearman tra le feature maggiormente correlate statisticamente con i compiti linguistici (vedi tabelle 3.3 e 3.4) e le feature con maggior peso secondo il modello di regressione lineare (vedi tabelle 4.2 e 4.3)

I due punteggi risultano molto simili e rivelano una discreta correlazione tra i due ranking, dimostrando così che il modello di regressione lineare è riuscito a individuare le feature che più correlano con i due task.

Infine, confrontando con una correlazione per ranghi di Spearman i due ranking di feature pesate prodotte dal modello di regressione lineare per i due task, si è ottenuto un punteggio di -0.65. Questo risultato (come già dimostrato precedentemente comparando i ranking delle feature più correlate con i task di accettabilità e complessità) suggerisce, ancora una volta, che i due compiti sono molto correlati e più una feature contribuisce a rendere una frase complessa, meno la rende accettabile.

4.2 Analisi con un modello neurale del linguaggio: BERT

Per capire quanto la presenza di feature linguistiche esplicite avesse impatto sulle performance del modello di regressione lineare SVR, successivamente, è stato addestrato un modello neurale del linguaggio (BERT) effettuando il *fine tuning* sui due task senza ricorrere, quindi, a feature linguistiche esplicite. Pertanto, è stata testata la capacità di BERT nell'esecuzione del task di regressione relativo alla predizione dei punteggi di *accettabilità* e *complessità* assegnati a ciascuna frase del dataset.
Come descritto precedentemente, infatti, a ciascuna frase, da parte di parlanti madrelingua, era stato assegnato un valore su una scala Likert da 1 a 7, dove il punteggio massimo indica, in un caso, che la frase è più accettabile e nell'altro, che è, invece, più complessa.

Per questa fase di esperimenti è stata utilizzata la versione preaddestrata di BERT disponibile su Hugging Face (in particolare è stato scelto il modello "dbmdz/bert-base-italian-uncased"[13] e il suo rispettivo *tokenizer*). I dati su cui è stato eseguito il pretraining del modello BERT italiano consistono in un insieme di testi recentemente scaricati da Wikipedia e una collezione del corpus OPUS. Il corpus di pretraining finale ha una dimensione di 13 GB e 2050057573 token.

Partendo dalla versione di BERT preaddestrata è stato eseguito il *fine tuning* rispettivamente sui task di *accettabilità* e *complessità*, utilizzando, anche questa volta

[13] https://huggingface.co/dbmdz/bert-base-italian-uncased

il metodo di *k-fold-crossvalidation* con un *k* uguale a 12. Il dataset di 672 frasi con i rispettivi giudizi è stato, perciò, suddiviso in 12 insiemi di 616 frasi di training e 56 di test.

Ciascuna frase è stata tokenizzata con il tokenizzatore messo a disposizione per il modello, ottenendo una suddivisione delle parole come quella riportata nell'esempio sottostante:

```
'[CLS]', 'ci', 'mostra', '##rono', 'il', 'paziente', 'che', 'l', "'", 'infermiera', 'voleva', 'chiamarlo', 'e', 'marco', 'medica', '##rlo', '.', 'SEP'
```

Come si può notare ciascuna frase è introdotta dalla parola chiave [CLS] e termina con la sequenza 'SEP'.

Ciascun training set è stato poi diviso in modo randomico in due sezioni, ottenendo il validation set. Le 616 frasi di training iniziale, quindi, sono state distribuite in 554 frasi di training e 62 di validation per ciascun *fold*.

Il fine tuning del modello è stato effettuato impostando un numero di epoche pari a 300. Questa scelta è stata effettuata sulla base dall'osservazione dell'andamento delle performance sul validation set durante la fase di training, nel tentativo di garantire sufficiente tempo alla rete di imparare dai dati forniti. Con un numero così elevato di epoche è risultata visibile una stabilizzazione delle performance attorno alla 250° epoca. Il modello selezionato per effettuare l'inferenza è quello ottenuto alla 300°.
Per quanto riguarda gli altri iperparametri è stato impostato il *weight decay* a 0.01 e il *learning rate* a 2e-5. Infine, come ottimizzatore è stato scelto *AdamW*[14].

Come metrica di valutazione delle performance è stato calcolata per ciascun fold la correlazione di Spearman tra i punteggi predetti dal modello per ciascuna frase e i valori reali.

[14] https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

La tabella 4.5 illustra le performance sui due task, ottenute sia sul validation che sul test set, facendo una media tra i valori di correlazione ottenuti per ciascun fold. Su ogni insieme, oltre alla media è riportata pure la deviazione standard.

			ACCETTABILITÀ	**COMPLESSITÀ**
SCORE	**VALIDATION SET**	*media*	0.778	0.776
		deviazione standard	0.037	0.036
	TEST SET	*media*	0.773	0.739
		deviazione standard	0.053	0.065

Tabella 4.5. Performance del *fine tuning* di BERT sui due compiti di analisi linguistica

Come si può notare mentre le performance ottenute sul validation set sono pressoché identiche tra i due diversi task, sul test set il modello neurale ottiene migliori prestazioni per il task dell'accettabilità.
Inoltre, in entrambi i casi, si nota un netto superamento rispetto ai punteggi ottenuti dal regressore lineare SVR. Infatti, considerando le performance del test set, nel caso dell'accettabilità si riscontra un miglioramento di ben 0.18 punti, nel caso della complessità, invece, c'è un aumento di 0.03 punti.

Come ulteriore metrica di valutazione per ciascun fold è stato calcolato l'errore quadratico medio (RMSE). Questa è una misura di errore assoluta in cui le deviazioni vengono elevate al quadrato per evitare che valori positivi e negativi possano annullarsi l'uno con l'altro. Formalmente è definita come segue:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\widehat{y_i} - y_i)^2}{n}}$$

Dove $\widehat{y_1}, \widehat{y_2}, \dots, \widehat{y_n}$ sono i valori predetti, mentre $y_1, y_2, \dots, y_n$ sono i valori osservati e *n* è il numero di osservazioni.
In generale indica l'adattamento assoluto del modello ai dati, dando come risultato quanto sono vicini i valori predetti rispetto a quelli reali. È una buona misura di quanto

accuratamente il modello predice l'inferenza e, più bassi sono i valori, migliore è la performance del modello.

La tabella 4.6 illustra i valori ottenuti su ciascuno dei 12 fold e una media di ciascun punteggio sia sul validation set che sul test set per i due task.

	ACCETTABILITÀ						**COMPLESSITÀ**					
	VALIDATION SET			**TEST SET**			**VALIDATION SET**			**TEST SET**		
FOLD	*RMSE*	*media*	*std*	*RMSE*	*media*	*std*	*RMSE*	*media*	*std*	*RMSE*	*media*	*std*
1-fold	1.20	1.22	0.01	1.21	1.22	0.04	0.96	0.99	0.02	0.99	0.96	0.04
2-fold	1.22			1.17			1.00			0.91		
3-fold	1.20			1.15			0.99			0.94		
4-fold	1.22			1.22			1.00			0.97		
5-fold	1.23			1.22			0.98			1.07		
6-fold	1.22			1.28			0.99			1.02		
7-fold	1.21			1.26			0.97			0.95		
8-fold	1.22			1.23			0.97			0.92		
9-fold	1.23			1.18			0.96			0.96		
10-fold	1.23			1.26			0.99			0.94		
11-fold	1.24			1.18			1.00			0.97		
12-fold	1.24			1.24			1.04			0.94		

Tabella 4.6. RMSE come metrica di valutazione per il *finetuning* di BERT

Come si può notare, con questa metrica il modello che ottiene performance migliori sia sul validation set che sul test set è quello testato sulla valutazione della *complessità* di una frase. I punteggi relativi ai due task risultano comunque molto vicini tra di loro, pertanto, la loro differenza non è molto rilevante ed entrambi i modelli ottengono buone prestazioni.

4.2.1 Transfer learning

Successivamente, sono stati elaborati due modelli che sfruttano tecniche di *transfer learning*, cioè che utilizzano la conoscenza precedentemente acquisita per la risoluzione di un problema, per affrontarne uno diverso ma correlato. In particolare, l'obiettivo di questo ulteriore studio è stato quello di comprendere se avere conoscenza su un determinato task può essere utile nella risoluzione di un altro. A tale scopo, infatti, per risolvere il compito di valutazione dell'accettabilità è stato sfruttato il

modello precedentemente messo a punto sulla valutazione della complessità e, viceversa, per affrontare il task della valutazione della complessità è stato utilizzato il modello su cui era stato fatto il *fine tuning* sul compito della valutazione dell'accettabilità. Anche per questi due modelli di *transfer learning* sono stati impostati gli stessi iperparametri descritti precedentemente ed è stata eseguita nuovamente una *k-fold-cross-validation* con k uguale a 12. Per ciascun fold è stato eseguito il *fine tuning* su un task e poi successivamente sull'altro (ad esempio se il modello doveva testare la complessità, veniva prima messo a punto sull'accettabilità su ciascun fold). Come metrica di valutazione delle performance dei modelli è stata usata ancora una volta la correlazione di Spearman tra i punteggi predetti e quelli reali.

La tabella 4.7. illustra i risultati ottenuti dai due nuovi modelli per entrambi i compiti, mostrando la media delle performance su tutti i fold sia per il validation set che per il test set.

			FINETUNING COMPLESSITÀ, ACCETTABILITÀ	**FINETUNING ACCETTABILITÀ, COMPLESSITÀ**
SCORE	**VALIDATION SET**	*media*	0.839	0.793
		deviazione standard	0.040	0.037
	TEST SET	*media*	0.852	0.764
		deviazione standard	0.037	0.064

Tabella 4.7. Performance del *fine tuning* di BERT sui due task con i due modelli di *transfer learning*

Come si può notare, anche in questo caso si ottengono performance superiori con il modello di *transfer learning* testato sulla valutazione dell'accettabilità come ultimo task. Infatti, sia sul validation set che sul test set si raggiungono punteggi più alti rispetto all'altro compito: nel primo caso si nota una discrepanza di 0.04 punti, nel secondo addirittura di 0.09. Precedentemente, con i due modelli direttamente messi a punto sui due task questa differenza tra le performance era meno accentuata ed era visibile solamente nel test set (con un punteggio di 0.77 per l'accettabilità e 0.74 per la complessità).

Inoltre, è interessante osservare come in entrambi i compiti ci sia un netto miglioramento con i modelli che sfruttano il *transfer learning*: infatti, rispetto ai modelli precedentemente addestrati si ottiene un aumento (rispettivamente per quanto riguarda l'accettabilità e la complessità) di 0.05 e 0.01 punti sul validation set e di e di 0.08 e 0.02 punti sul test set. Questo risultato suggerisce che avere precedentemente acquisito conoscenza su un altro task (accettabilità o complessità) rispetto a quello su cui deve fare le predizioni finali è di grande aiuto per il modello, probabilmente perché i due compiti sono molto simili e, infatti, come già più volte descritto, molto correlati (con un punteggio di -0.49).

Nel prossimo capitolo verrà illustrata un'analisi approfondita dei quattro modelli elaborati (quelli con cui è stato eseguito il *fine tuning* sui due task e quelli che sfruttano tecniche di *transfer learning* per risolvere i due compiti) per comprendere meglio come immagazzinano l'informazione linguistica e costruiscono le rappresentazioni di una frase. Oltre a un'analisi specifica per ciascun modello, verrà fornito anche un confronto tra questi.

5. Studio sulle rappresentazioni di BERT

In questo capitolo verrà illustrata un'analisi dei quattro modelli descritti precedentemente (più in particolare: i due direttamente testati sui due compiti linguistici di *accettabilità* e *complessità* e i due che sfruttano tecniche di *transfer learning*). In primo luogo, verranno estratte diverse rappresentazioni delle frasi del dataset IUDT (descritto nel capitolo 3.2.2) da ciascuno strato di ogni modello e poi valutata la similarità cosenica tra queste. Successivamente verrà descritta una profilazione linguistica dei modelli neurali elaborati, soffermandosi su come cambiano le loro competenze lungo i diversi layer.

5.1 Similarità tra le rappresentazioni dei modelli neurali

Al fine di valutare la similarità tra le rappresentazioni delle frasi estratte dai modelli neurali, per prima cosa è stato eseguito, per ciascun modello, il *fine tuning* sull'intero dataset, per avere a disposizione maggior conoscenza possibile. Dopo aver completato il *fine tuning* e aver memorizzato i modelli messi a punto sui diversi task, sono state estratte le rappresentazioni (layer per layer) delle 35480 frasi del dataset IUDT con ciascun modello elaborato.

Più in particolare sono stati utilizzati i seguenti modelli:

1. BERT preaddestrato;
2. BERT con *fine tuning* sul compito linguistico dell'accettabilità;
3. BERT con *fine tuning* sul compito linguistico della complessità;
4. BERT con *fine tuning* prima sul compito dell'accettabilità e poi della complessità (utilizzando il *transfer learning*);
5. BERT con *fine tuning* prima sul compito della complessità e poi dell'accettabilità (utilizzando il *transfer learning*);

In questo modo si sono ottenute un totale di 2128800 rappresentazioni, ossia 35480 per i cinque modelli appena elencati, per ciascuno dei 12 layer di ciascun modello.

Successivamente, è stata calcolata la similarità cosenica tra le diverse rappresentazioni di ciascun layer, facendo una media ad ogni strato del modello. Il valore della similarità

cosenica è stato ottenuto tramite la funzione *cosine_similarity*[15] di *SciKitLearn*. Questa calcola tale valore come il *dot product* normalizzato di due vettori di rappresentazioni X e Y:

$$K(X,Y) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|}$$

Questa equazione può restituire un risultato che va da 0 a 1. Più in particolare, restituisce 0 se i vettori delle rappresentazioni sono completamente diversi tra loro e 1 quando invece si sovrappongono.

La figura 5.1 illustra i risultati ottenuti. Più in particolare è stata valutata la similarità tra:

1. Le rappresentazioni estratte dal modello BERT preaddestrato e le rappresentazioni di BERT con *fine tuning* sul compito di valutazione dell'accettabilità (in figura 5.1 indicato come *ud_pretVSud_f_acc*, con la linea rossa);
2. Le rappresentazioni estratte dal modello BERT preaddestrato e le rappresentazioni di BERT con *fine tuning* sul compito di valutazione della complessità (in figura 5.1 indicato come *ud_pretVSud_f_compl*, con la linea viola);
3. Le rappresentazioni estratte dal modello BERT con *fine tuning* sul compito di valutazione della accettabilità e le rappresentazioni di BERT con *fine tuning* sul compito di valutazione della complessità (in figura 5.1 indicato come *f_accVSf_compl*, con la linea arancione);
4. Le rappresentazioni estratte dal modello BERT messo a punto prima sul compito di valutazione dell'accettabilità e poi della complessità (sfruttando il *transfer learning*) e le rappresentazioni di BERT con *fine tuning* sul compito di valutazione della complessità (in figura 5.1 indicato come *f_acc_complVSf_compl*, con la linea celeste);
5. Le rappresentazioni estratte dal modello BERT messo a punto prima sul compito di valutazione della complessità e poi della accettabilità (sfruttando tecniche di *transfer learning*) e le rappresentazioni di BERT con *fine tuning*

[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

sul compito di valutazione dell'accettabilità (in figura 5.1 indicato come *f_compl_accVSf_acc*, con la linea verde);

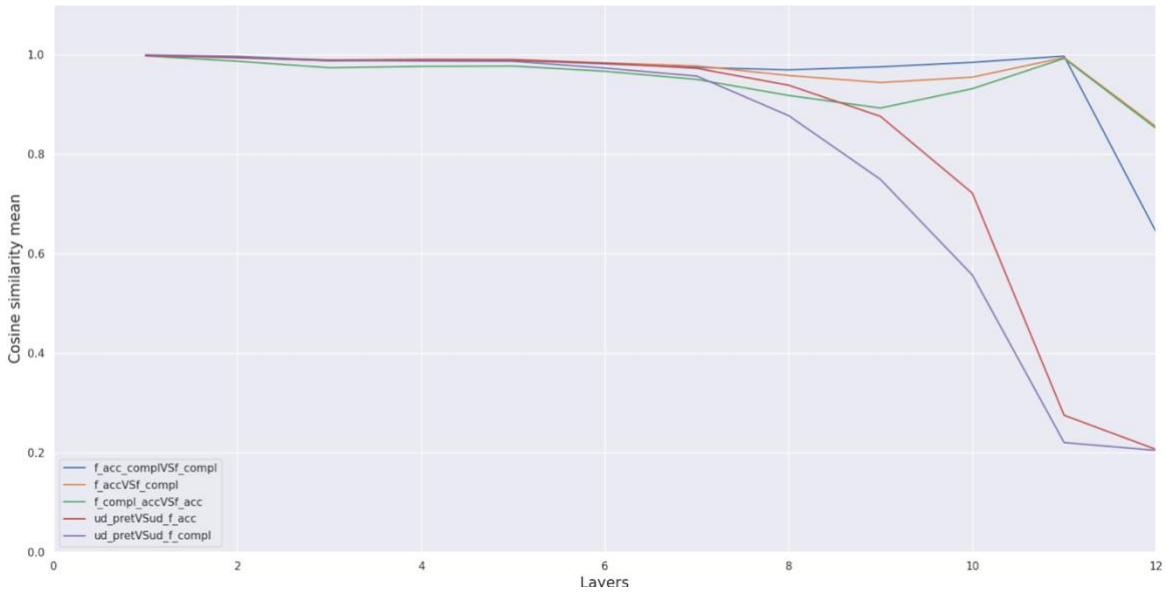


Figura 5.1. Similarità cosenica delle rappresentazioni dei 5 modelli, layer per layer

Come si può notare, dai confronti tra tutti i modelli si ottengono rappresentazioni molto simili fino al settimo layer. A partire dall'ottavo strato, invece, è ben visibile un calo del punteggio di similarità cosenica per quanto riguarda il confronto delle rappresentazioni di BERT preaddestrato sia con quelle di BERT messo a punto sull'accettabilità che sulla complessità. È interessante notare che questa diminuzione nel punteggio di similarità è più marcata nel confronto tra le rappresentazioni di BERT preaddestrato e le rappresentazioni ottenute dopo il *fine tuning* con la complessità, infatti (in figura 5.1) la linea viola si posiziona sempre sotto quella rossa (che invece è relativa al confronto con le rappresentazioni ottenute dopo il *fine tuning* con l'accettabilità). In ogni caso, sempre per quanto riguarda questi due confronti, il punteggio di similarità ottenuto nel dodicesimo layer è lo stesso per entrambi e si aggira intorno allo 0.2.

Relativamente agli altri modelli, è degno di nota l'andamento della similarità cosenica tra le rappresentazioni ottenute dopo il *fine tuning* dell'accettabilità e della complessità e quelle ottenute dopo solamente il *fine tuning* della complessità. I punteggi relativi a questo confronto, fino all'undicesimo layer, sono quelli più elevati. Al dodicesimo layer, invece, si verifica un calo di circa 0.3 punti, ottenendo così una similarità cosenica prossima allo 0.7.

Inoltre, sia per il confronto tra le rappresentazioni ottenute dopo il *fine tuning* con l'accettabilità e quelle dopo il *fine tuning* con la complessità, che per quelle dopo il *fine tuning* con la complessità e l'accettabilità confrontate con il *fine tuning* diretto sull'accettabilità, si nota un calo (più marcato nel secondo confronto) intorno al nono layer, seguito da una successiva ripresa fino all'undicesimo layer, per poi diminuire nuovamente al dodicesimo, raggiungendo medie simili intorno allo 0.85.

Si può quindi concludere che i confronti con punteggi di similarità cosenica superiori sono quelli in cui si estraggono rappresentazioni da modelli messi a punto sui diversi compiti linguistici. L'alta somiglianza tra le rappresentazioni ottenute dopo il *fine tuning* diretto sui due task (linea arancione in figura 5.1) suggerisce che BERT, seppur addestrato su due compiti linguistici diversi, immagazzina, nella maggior parte degli strati, l'informazione in modo simile, con una lieve discrepanza nell'ultimo layer (in cui avviene la specializzazione specifica per la risoluzione del compito). Questo, ancora una volta, è una conseguenza dell'alta correlazione dei diversi compiti linguistici, che quindi spingono i modelli a carpire informazioni simili per la loro esecuzione.

Infine, per quanto riguarda i due modelli di *transfer learning*, si ottengono rappresentazioni più simili tra loro in modelli prima messi a punto sulla complessità e l'accettabilità confrontati con modelli direttamente addestrati sull'accettabilità, rispetto a quelli con *fine tuning* prima sull'accettabilità e poi la complessità confrontati con i modelli direttamente addestrati sulla complessità. Questo suggerisce che un primo *fine tuning* sull'accettabilità (seguito da uno sulla complessità) porta un'importante variazione nella costruzione delle rappresentazioni, mentre un primo *fine tuning* sulla complessità (e poi uno successivo sull'accettabilità) non introduce un altrettanto divario marcato nella costruzione delle rappresentazioni (se rapportate a quelle ottenute direttamente da un modello addestrato sull'accettabilità).

In ogni caso questa differenza, introdotta dall'aver addestrato prima il modello su un altro task, contiene informazione utile, in quanto (come descritto nel capitolo 4.2) si ottengono performance superiori con i modelli di *transfer* learning. In particolare, rispetto ai modelli con *fine tuning* diretto sui due compiti linguistici, si guadagnano 0.2 punti nella correlazione di Spearman con il modello addestrato prima

sull'accettabilità e poi sulla complessità e 0.8 punti con il modello prima messo a punto sulla complessità e poi sull'accettabilità.

5.2 Profilazione linguistica dei modelli neurali

Per capire come varia la conoscenza linguistica codificata in BERT prima e dopo il processo di *fine tuning* sui diversi compiti linguistici e con l'utilizzo o meno di metodi di *transfer learning*, sono stati eseguiti 82 *probing task*, ciascuno corrispondente a feature che catturano lessico, morfosintassi e sintassi di una frase.
Tali *probing task* sono stati effettuati usando le rappresentazioni contestuali di ciascun modello per ogni frase del dataset IUDT e predicendo il valore reale di ciascuna feature linguistica lungo i layer interni. In questo modo, sono state individuate variazioni nell'informazione linguistica codificata tra i diversi modelli. Questo modello di *probing* è stato ripreso da quello descritto nel capitolo 2.2.

Le feature su cui sono stati eseguiti i *probing* sono state estratte direttamente dalla piattaforma *Profiling UD* e possono essere categorizzate in nove gruppi corrispondenti a diversi fenomeni linguistici (tabella 5.1):

Livello di annotazione	**Gruppi di feature**	**Singole feature**
Raw Text	*Raw Text*	*sent_length*
		char_per_tok
Vocabolario	*Ricchezza del vocabolario (Vocabulary)*	*ttr_form, ttr_lemma*
POS tagging	*Informazione morfosintattica (POS)*	*upos_dist*, xpos_dist**
		lexical_density
	Morfologia inflessionale (VerbInflection)	*aux**
	Struttura verbale del predicato (VerbPredicate)	*verbal_head_per_sent, verbal_root_perc, avg_verb_edges*

Dependency Parsing	*Tree Structure*	*avg_token_per_clause, avg_links_len, max_links_len, avg_max_depth, avg_prepositional_chain_len, prep_dist**
	Ordine degli elementi (Order)	*subj_pre, obj_post*
	Relazioni sintattiche (SyntaticDep)	*dep_dist**
	Uso delle subordinate (Subord)	*subordinate_dist_1, avg_subordinate_chain_len, principal_proposition_dist, subordinate_proposition_dist, subordinate_post*

Tabella 5.1. Gruppi di feature utilizzate per i probing task

La divisione in questi gruppi di feature è stata scelta prendendo come spunto il lavoro di Miaschi, 2020. Una descrizione approfondita di ciascuna feature è presente in appendice (capitolo 9.1).

Più nello specifico, per questa sperimentazione sono state utilizzate le rappresentazioni delle frasi del dataset IUDT estratte dai cinque modelli elaborati (BERT *preaddestrato*, i due modelli di BERT con *finetuning* sui due compiti linguistici e i due modelli di *transfer learning*), ottenendo una rappresentazione per ciascun layer. A ciascuna di queste rappresentazioni sono stati associati i valori delle feature della frase corrispondente. I punteggi di ciascuna caratteristica linguistica, come detto precedentemente, sono stati acquisiti tramite la piattaforma *Profiling UD*. Successivamente, per il *probing*, è stato utilizzato un *LinearSVR* (*linear support vector regression*) al fine di predire i valori di ciascuna feature. Il *LinearSVR* è stato addestrato considerando 25480 rappresentazioni (con i relativi valori delle caratteristiche linguistiche) come training e 10000 come test. La capacità di predizione di ciascuna feature è stata valutata tramite la correlazione di Spearman tra il valore reale e il valore predetto.

Per valutare quali feature sono più significative, è stata calcolata una *baseline* per ciascuna caratteristica linguistica, dando in pasto al *LinearSVR* l'indicazione del numero di token della frase (anziché la rappresentazione) e, in fase di addestramento, i valori delle caratteristiche linguistiche. In questo modo, se la predizione delle feature (valutata nuovamente tramite una correlazione di Spearman) è migliore rispetto all'utilizzo delle rappresentazioni di BERT, significa che BERT non ha contribuito in alcun modo all'informazione necessaria per quella feature.

Confrontando i valori della baseline con quelli ottenuti nel dodicesimo layer, su 82 *feature*, 10 sono risultate al di sotto di essa per quanto riguarda il modello direttamente messo a punto sul compito dell'accettabilità e per quello prima addestrato sull'accettabilità e poi sulla complessità. Gli altri modelli, invece, hanno riscontrato 9, 11 e 8 feature al di sotto della baseline (rispettivamente: il modello con *fine tuning* sulla complessità e accettabilità, il modello di BERT *preaddestrato* e il modello con *fine tuning* diretto sulla complessità come task finale). Di seguito (nelle figure 5.3 – 5.7) verranno descritti i risultati ottenuti, prendendo in considerazione le *feature* con punteggi al di sopra della baseline.

La figura 5.2 mostra quanto emerso dai probing task sul modello di BERT preaddestrato. La colonna B riporta i valori della baseline per ciascuna *feature*.
Come si può notare, questo modello, nonostante non sia stato messo a punto su nessun compito specifico, ha molta competenza sulle caratteristiche linguistiche relative alle informazioni di base, come la lunghezza della frase o la distribuzione della punteggiatura.
Inoltre, si mostra capace nella predizione delle informazioni relative all'altezza dell'albero sintattico e alle catene proposizionali (*n_propositional_chains*). Le caratteristiche linguistiche su cui, invece, è più in difficoltà sono quelle riguardanti la valenza verbale (*verbal_edges_dist_**).

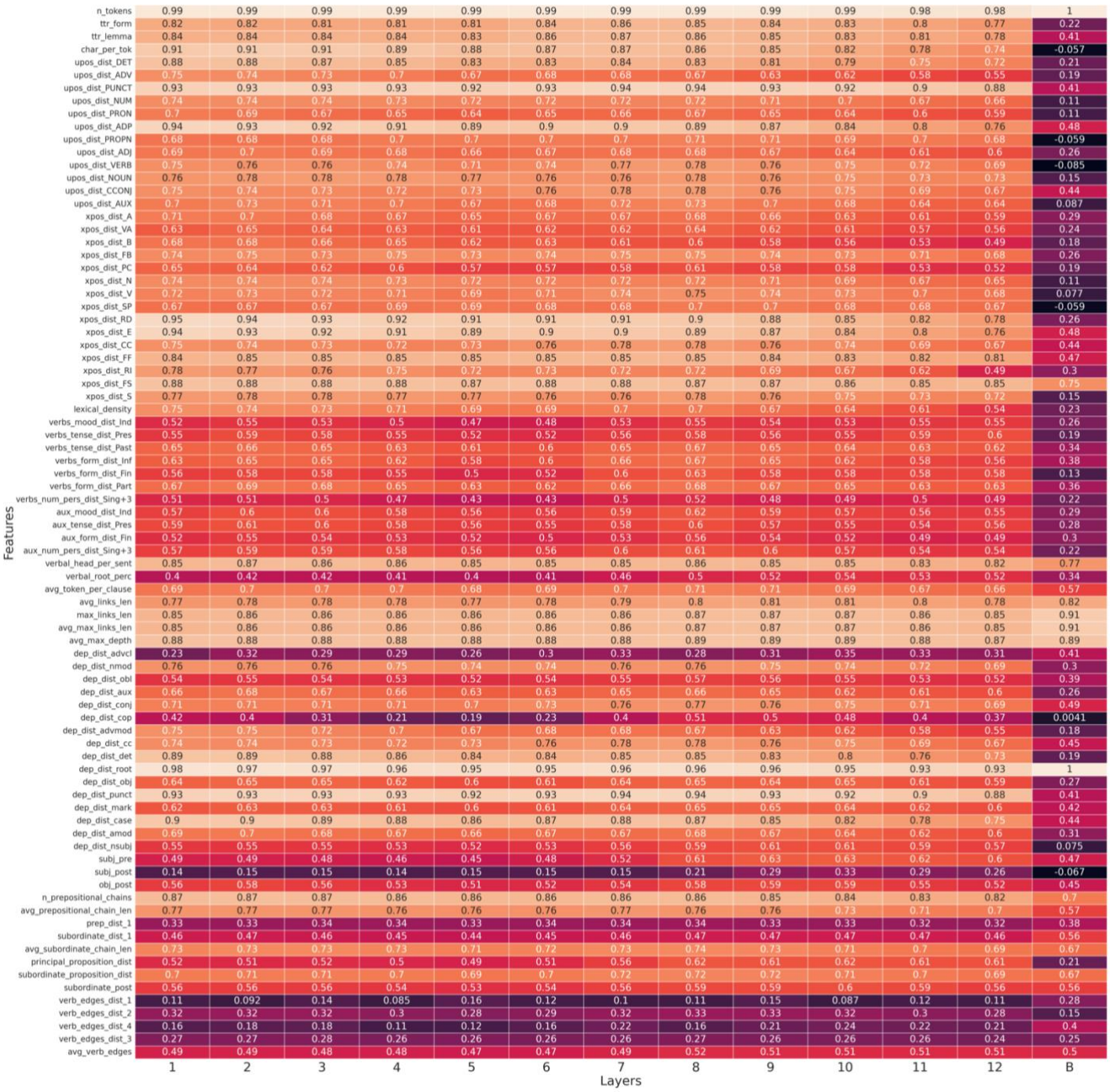


Figura 5.2. Risultati dei *probing task* sul modello di BERT preaddestrato, feature per feature

Successivamente, le caratteristiche linguistiche utilizzate sono state raggruppate (così come descritto in tabella 5.1) in nove gruppi (*RawText*, *Vocabulary*, *POS*, *VerbInflection*, *VerbPredicate*, *TreeStructure*, *Order*, *SyntaticDep*, *Subord*). Di seguito (figura 5.3), si riportano i risultati suddivisi in insiemi di *feature* per una migliore visualizzazione degli andamenti lungo gli strati del modello preaddestrato.

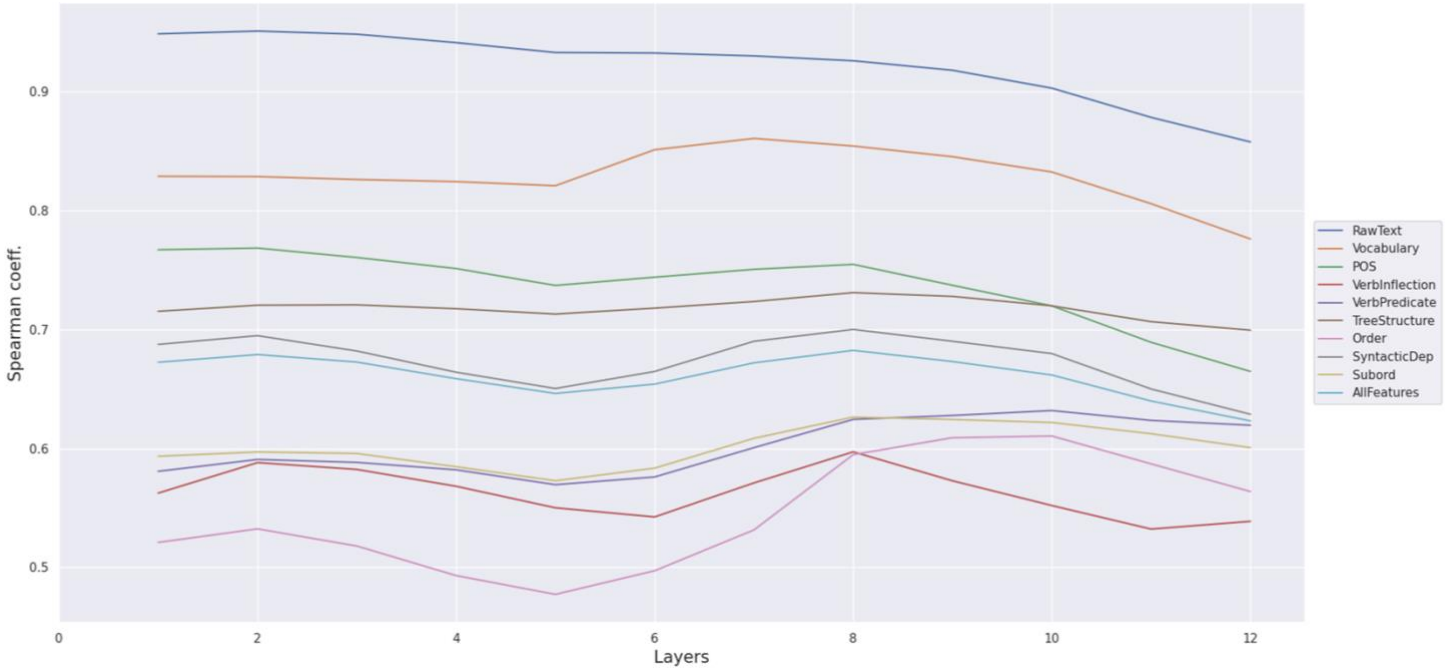


Figura 5.3. Variazione dell'informazione linguistica lungo i layer del modello BERT preaddestrato

La linea celeste (*AllFeatures*) illustra la tendenza media di ogni *feature* lungo i dodici strati. In generale, si nota un orientamento decrescente: mentre nel primo layer la media dei punteggi è di 0.672, nell'ultimo è di 0.623. Tale punteggio medio deriva dal fatto che quasi tutti i gruppi di feature ottengono punteggi più bassi negli ultimi layer, mostrando un declino graduale lungo gli strati (alternati, talvolta a picchi di ripresa). È il caso, ad esempio, del gruppo relativo alle *POS*, di quello riguardante le relazioni sintattiche (*SyntacticDep*), di quello della morfologia inflessionale (*VerbInflection*) e dei primi insiemi di *feature* (*RawText* e *Vocabulary*). Un caso analogo è, inoltre, dato dall'insieme delle caratteristiche relative alla struttura ad albero, mostrando, tuttavia, un declino meno marcato. Presentano invece un deterioramento, seguito da un miglioramento evidente negli ultimi layer gli insiemi *Order*, *Subord* e *VerbPredicate*. Inoltre, il gruppo di feature che ottiene performance migliori è quello relativo alle caratteristiche *raw* del testo. Questo insieme raggiunge punteggi superiori rispetto agli altri lungo tutti i dodici i layer, mostrando, tuttavia, un leggero deterioramento negli strati finali. Per quanto riguarda le caratteristiche linguistiche inerenti al vocabolario, è interessante notare un miglioramento delle performance intorno al sesto layer, seguito da un peggioramento negli strati successivi.

La figura 5.4, invece, illustra i risultati ottenuti sui probing task per i modelli di BERT: 1. con *fine tuning* diretto sull'accettabilità o 2. sulla complessità, 3. con *transfer learning*, essendo prima addestrato sulla complessità e poi sull'accettabilità o 4. essendo prima addestrato sull'accettabilità e poi sulla complessità. Anche in questo caso, le *feature* sono state raggruppate seguendo la suddivisione illustrata nella tabella 5.1.

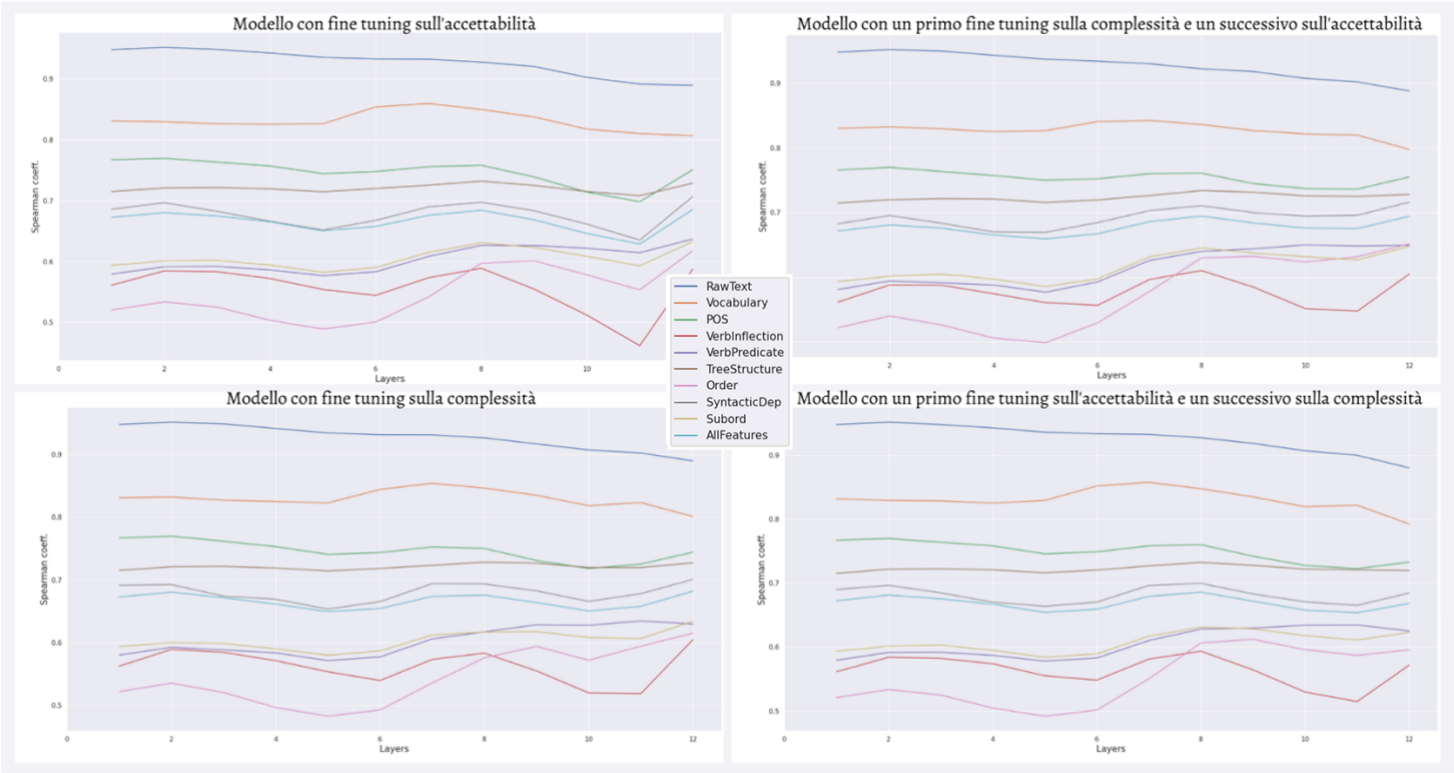


Figura 5.4. Variazione dell'informazione linguistica lungo i layer nei modelli messi a punto sui compiti linguistici di accettabilità e complessità

Dalla figura 5.4 si evince che:

1. Anche in questi quattro grafici, i gruppi di *feature* che ottengono performance migliori sono quello relativo alle caratteristiche *raw text* e al vocabolario, esattamente come accadeva nel modello preaddestrato (figura 5.3). Questo probabilmente è dovuto al fatto che sono caratteristiche linguistiche di base e, quindi, facilmente predicibili anche senza uno specifico *fine tuning* su un task;
2. Rispetto a quanto si osservava nella figura 5.3, in cui quasi tutti gli insiemi di *feature* ottengono punteggi più bassi negli ultimi layer, in questi grafici (tralasciando i gruppi *RawText* e *Vocabulary*) si verifica quasi sempre un

miglioramento nei layer di output rispetto ai precedenti strati. Tale tendenza è evidente nella linea celeste (*AllFeatures*) di ciascun grafico, in cui è riportata la media di ogni gruppo di caratteristiche linguistiche. Più nello specifico, nel layer di output di *AllFeatures*, si registra un punteggio di 0.68 contro lo 0.67 del primo strato per i modelli messi a punto sull'accettabilità e sulla complessità. Per i modelli di *transfer learning*, sempre guardando *AllFeatures*, mentre non vi è una variazione di punteggio tra il primo e ultimo layer (0.67 in entrambi i casi) nel modello con un primo *fine tuning* sull'accettabilità e un secondo sulla complessità, si nota un incremento di 0.02 punti nello strato di output (0.69) del modello con *fine tuning* sulla complessità e l'accettabilità.
Inoltre, è interessante notare che, in tutti i casi, nel layer di output, si verifica un miglioramento dei punteggi dei modelli con *fine tuning* sui compiti linguistici rispetto al modello preaddestrato (in cui si registrava 0.62). Tale incremento indica che i compiti su cui sono addestrati i modelli sono prettamente linguistici;

3. Più in particolare, (sempre non tenendo in considerazione gli insiemi *RawText* e *Vocabulary*) tutti i gruppi di *feature* tendono a peggiorare intorno al decimo/undicesimo layer per poi avere una netta ripresa al dodicesimo (il layer di output). Questa tendenza è significativa perché mostra come i modelli sviluppino l'informazione necessaria per la risoluzione dei compiti finali;
4. Approfondendo quanto detto al punto precedente, mentre il modello con *fine tuning* sull'accettabilità, generalmente, ottiene (per la maggior parte dei gruppi di *feature*) una riduzione del valore di correlazione di Spearman intorno all'undicesimo layer, mentre nel grafico del modello con *fine tuning* sulla complessità, ciò avviene al decimo;
5. Più in generale, sempre tralasciando le prime due linee, la tendenza dei quattro grafici è molto simile (anche se in alcuni di questi è più marcata): dopo un lieve picco nel secondo strato, si ha un declino tra il quinto e il sesto, seguito da una ripresa (che raggiunge l'apice intorno all'ottavo layer), una perdita (come già descritto nel terzo punto) intorno al decimo/undicesimo layer e un miglioramento nel dodicesimo;
6. In tutti e quattro i grafici, il gruppo di *feature* che mostra più variazione è quello relativo all'inflessione verbale (linea rossa) e rosa. Più nello specifico si ottiene

un calo netto più marcato per quanto riguarda il modello con *fine tuning* sull'accettabilità;

Dopodiché, per approfondire meglio come varia la competenza linguistica di ciascuna *feature* dopo il processo di *fine tuning* è stato calcolato il *gain*, ossia la percentuale di conoscenza acquisita. Più nello specifico, si è calcolata la percentuale di guadagno (in ciascuna feature) dei modelli con *fine tuning* sui diversi compiti linguistici rispetto al semplice modello BERT preaddestrato, attraverso la seguente formula:

$$Gain = \frac{feature_{modelloFinetuned} - feature_{modelloPreddestrato}}{feature_{modelloPreaddestrato}}$$

La figura 5.5 illustra i risultati ottenuti nei vari gruppi di feature, confrontando i modelli con *fine tuning* diretto sui due compiti linguistici e il modello BERT preaddestrato.

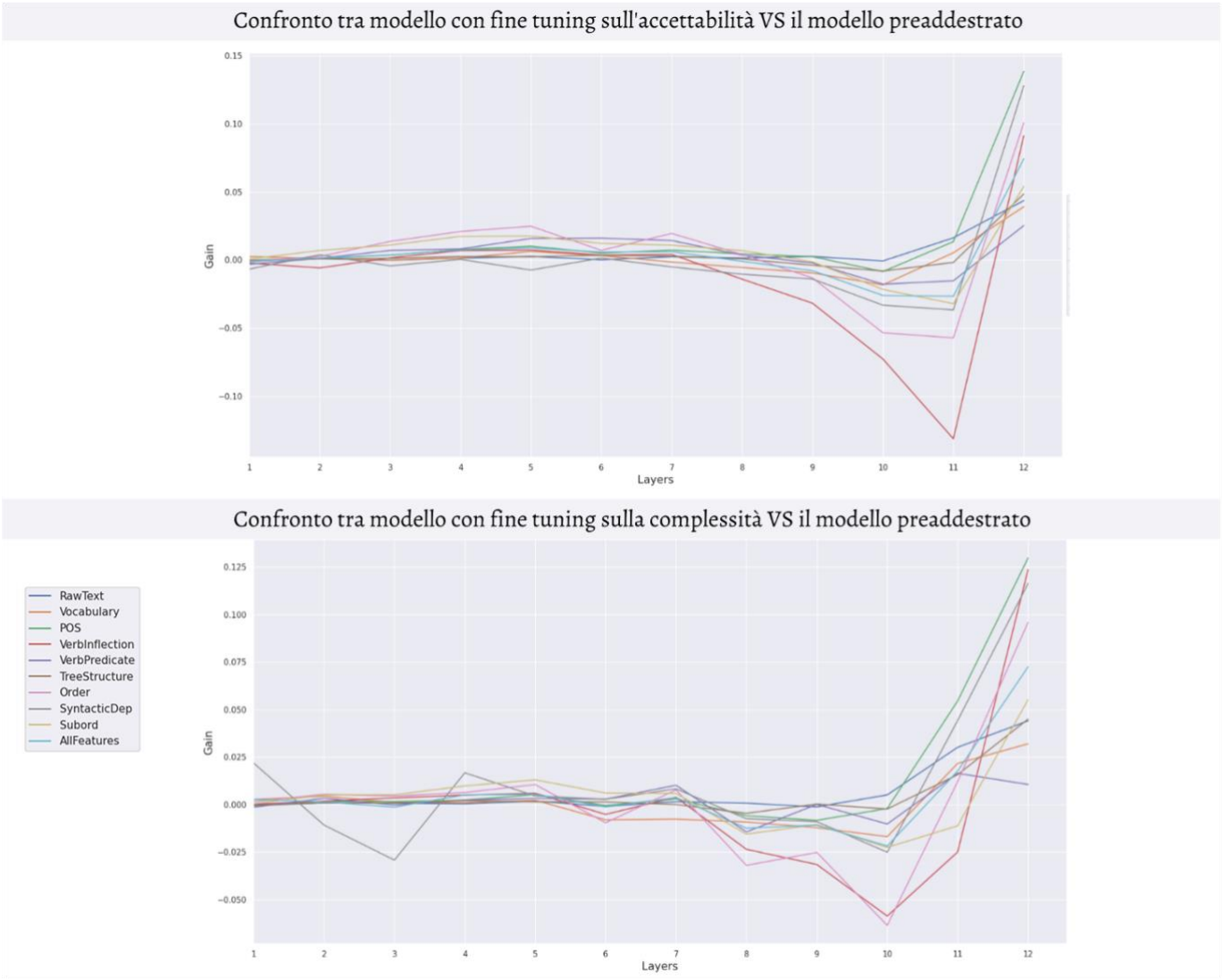


Figura 5.5. Gain tra modelli con *fine tuning* diretto sui singoli task e modello preaddestrato

Nella figura 5.5 si riscontra, in entrambi i casi, un miglioramento delle competenze di ciascun gruppo di *feature* nel layer di output. Più nello specifico, nei due grafici, mentre nell'ultimo layer le caratteristiche linguistiche relative alle parti del discorso ottengono il maggior guadagno percentuale della conoscenza (con un aumento superiore al 10%), le caratteristiche relative al predicato verbale risultano quelle con minor incremento (circa l'1-2%). Inoltre, il peggioramento delle tendenze (già visibile nel grafico 5.4) al decimo strato (nel caso del modello con *fine tuning* sulla complessità) e all'undicesimo (nel caso del modello con *fine tuning* sull'accettabilità) è in questa figura più marcato. Per di più, come si era già notato nel caso del modello addestrato direttamente sull'accettabilità, la maggior perdita si ha per il gruppo di feature relativo all'inflessione verbale. Il picco minimo, invece, nel caso del modello con *fine tuning* sulla complessità è dato dalle *feature* relative all'ordine.

La figura 5.6 mostra i risultati ottenuti nei vari gruppi di feature, confrontando i modelli di *transfer learning* e il modello BERT preaddestrato.

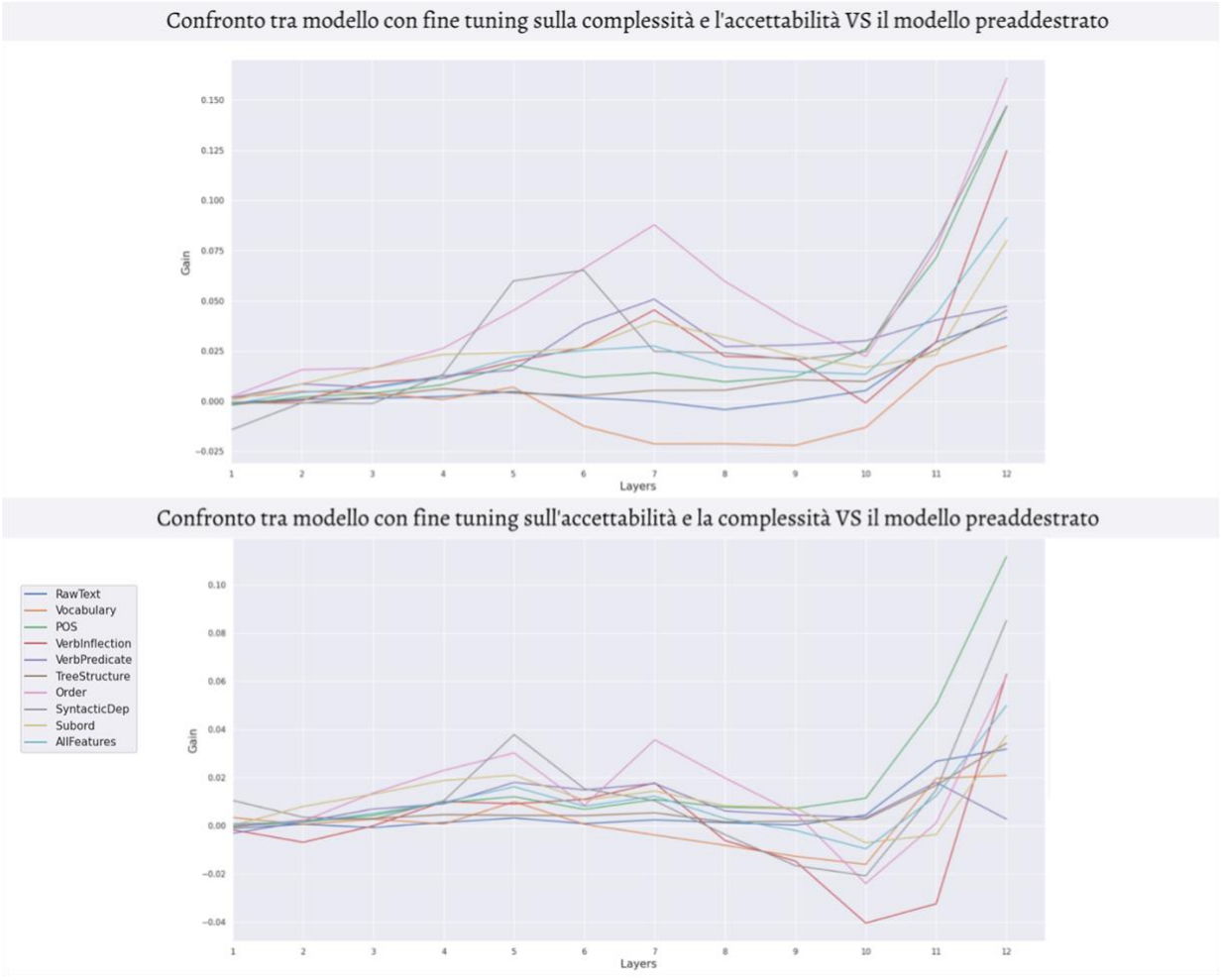


Figura 5.6. Gain tra modelli di *transfer learning* e modello preaddestrato

È interessante notare che rispetto alla figura 5.5, vi sono variazioni più marcate nei gruppi *feature*. Infatti, sono ben visibili diversi picchi in entrambi i confronti. Nel caso del primo grafico, il più evidente risulta quello intorno al settimo layer, relativo alle caratteristiche linguistiche riguardanti l'ordine (8%). Tra l'altro, in questo caso è proprio questo insieme di *feature* a raggiungere il maggior incremento delle competenze nel layer di output.
Nel secondo grafico, invece, i due picchi più evidenti sono dati da *Order* (sempre nel settimo layer) con una percentuale minore rispetto al grafico descritto precedentemente (3.7%) e da *SyntacticDep* al quinto layer (3.8%). Inoltre, in questo grafico, il gruppo di feature che ottiene il maggior incremento delle competenze linguistiche è quello relativo alle parti del discorso.

Un'ulteriore differenza tra i due grafici in figura (5.6) è data dai gruppi di *feature* che ottengono i valori minimi nel layer di output: mentre nel primo caso il gruppo con punteggio più basso è *Subord* (che risulta avere punteggi minori sin dal sesto layer), nel secondo caso è *VerbPredicate*. È interessante, però, notare che, in questo secondo confronto, considerando tutti gli strati, il minimo è raggiunto da *VerbInflection* nel decimo layer.

Successivamente, per comprendere la variazione di competenza linguistica di ciascuna *feature* nei modelli di *transfer learning* rispetto ai modelli addestrati direttamente sui due compiti linguistici, è stata applicata la seguente formula:

$$Gain = \frac{feature_{modelloTransferLearning} - feature_{modello}}{feature_{modello}}$$

La figura 5.7 illustra i risultati ottenuti nei vari gruppi di feature, confrontando i modelli di transfer learning con i modelli direttamente messi a punto sui singoli compiti linguistici.

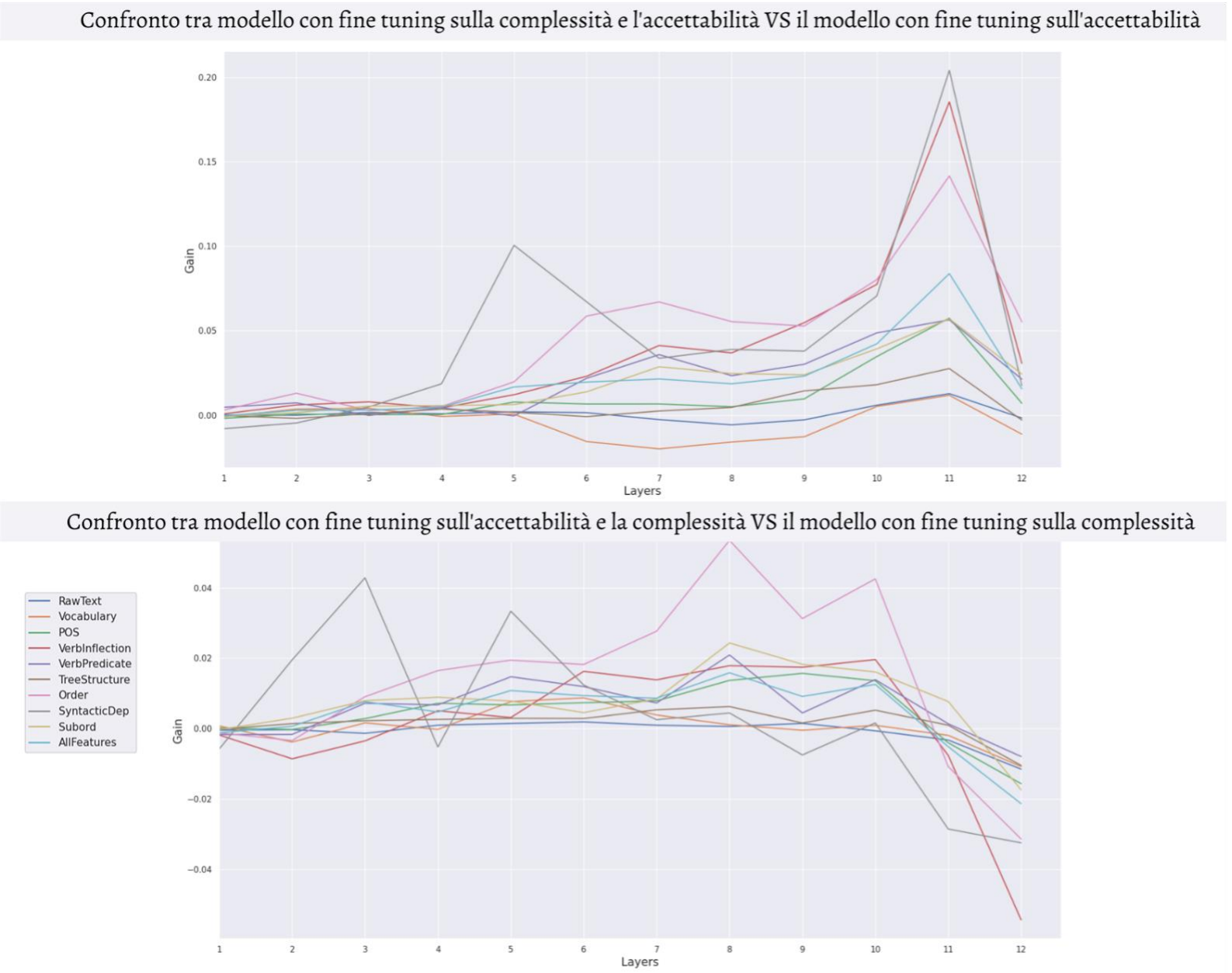


Figura 5.7. Gain tra i modelli di *transfer learning* e i modelli direttamente messi a punto sui singoli task

Dalla figura 5.7 emerge che:

- Nel layer di output, mentre nel primo grafico la maggior parte delle caratteristiche linguistiche ottiene punteggi superiori allo zero, nel secondo tutte le *feature* risultano inferiori. Ciò significa che, facendo *fine tuning* prima sull'accettabilità e poi sulla complessità, rispetto a quanto accade con un *fine tuning* diretto sulla complessità, vi è una perdita di competenze in ciascun gruppo di caratteristiche linguistiche;

- Sebbene nel secondo grafico i punteggi di tutti i gruppi di *feature* siano inferiori allo 0 (nel layer di output), la perdita riscontrata non è molto forte (il punteggio più basso è -4.5% per il gruppo *VerbInflection*, mentre il più alto è circa -1% per il gruppo *VerbPredicate*);
- Nel primo grafico, l'insieme di caratteristiche linguistiche in cui si riscontra un guadagno della competenza maggiore (nell'ultimo layer) è dato da *Order*, così come avveniva nel confronto tra il modello con *fine tuning* sulla complessità e l'accettabilità vs il modello preaddestrato. Ma, mentre in quel caso si otteneva un miglioramento del 16%, in questo grafico del 6%.

  È interessante, inoltre, notare che questi due confronti, oltre ad avere lo stesso gruppo di *feature* con punteggi migliori, ha anche lo stesso insieme con punteggi più bassi: infatti, in entrambi i casi la tendenza che nel layer di output ottiene performance minori è data da *Vocabulary*;
- In entrambi i grafici in figura, si osservano picchi intorno all'undicesimo (in un caso) e al decimo layer (nell'altro). Questo rispecchia quanto visto in precedenza in figura 5.4. Infatti, il modello con *fine tuning* diretto sull'accettabilità aveva un declino intorno all'undicesimo layer e quello messo a punto direttamente sulla complessità aveva un peggioramento intorno al decimo. Essendo tale tendenza meno marcata nei gruppi di feature dei modelli di transfer learning, si ottengono così, in questi grafici, picchi intorno a questi layer;

Infine, come ulteriore analisi statistica, è stato estratto, dal layer di output, il ranking delle caratteristiche linguistiche con maggior gain (ottenuto confrontando i modelli con *fine tuning* diretto sui singoli task e il modello preaddestrato) ed è stata calcolata la correlazione di Spearman tra i due ranking e tra questi sia con le *feature* più importanti per il regressore lineare (descritto nel capitolo 4.1), che con le caratteristiche linguistiche maggiormente correlate con l'*accettabilità* e la *complessità* (descritte nel capitolo 3.2.1).

Prima di illustrare i vari punteggi di correlazione, si riportano, per completezza, le *heatmap* (figura 5.8 e figura 5.9) relative al *gain* tra i modelli messi a punto sui due compiti linguistici e il modello preaddestrato. In ciascuna delle due figure, è inserita come ultima riga la media dei gain per ciascun *layer*.

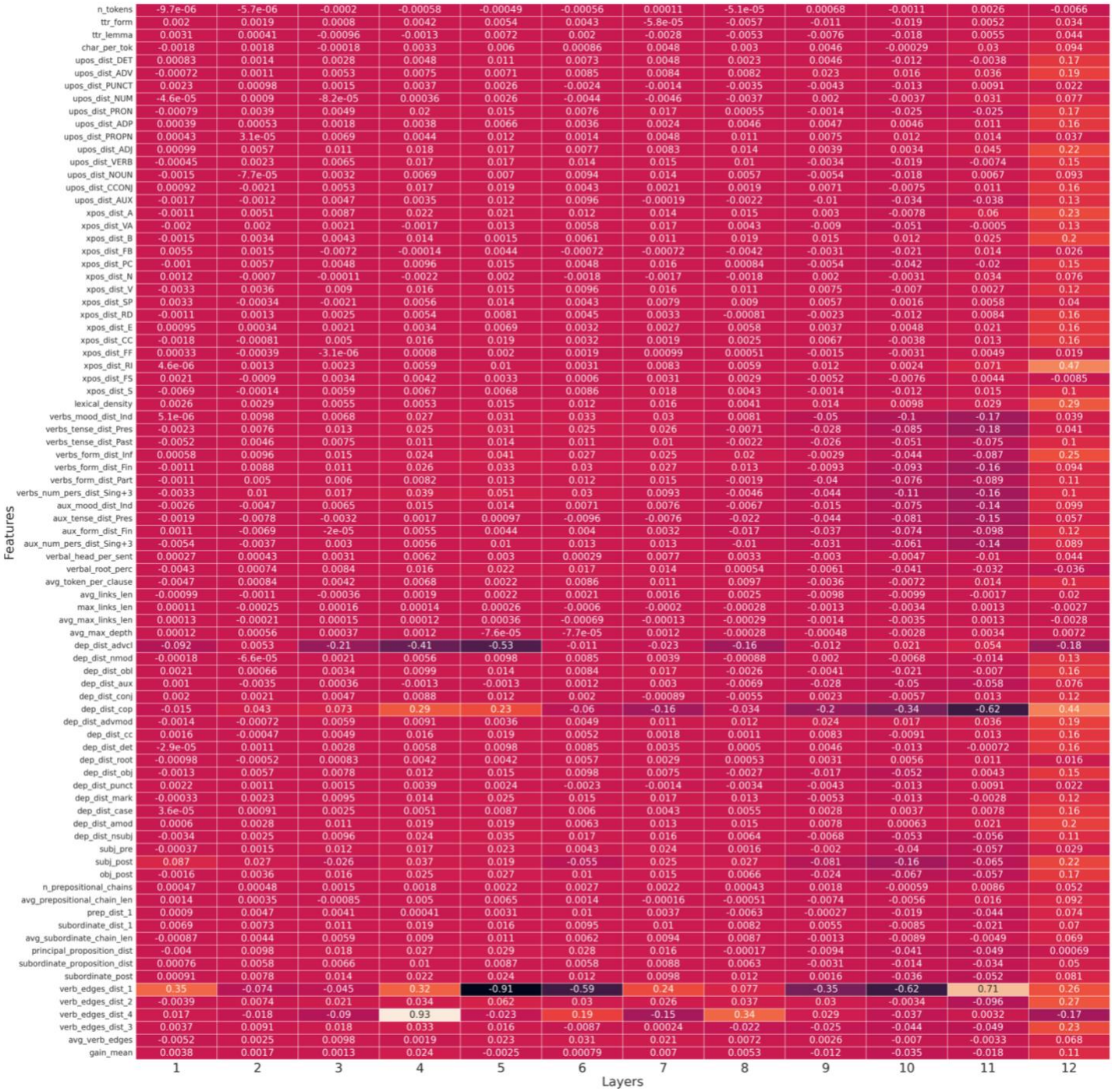


Figura 5.8. Heatmap con calcolo del gain tra modello con *fine tuning* sull'accettabilità e modello preaddestrato

Si può notare che, per quanto riguarda il confronto tra il modello con *fine tuning* sull'accettabilità e il modello preaddestrato, nell'ultimo layer, le *feature* con punteggio maggiore nel gain sono *xpos_dist_RI*, *dep_dist_cop*, *lexical density*, *verb_edges_dist_**, *xpos_dist_A*.

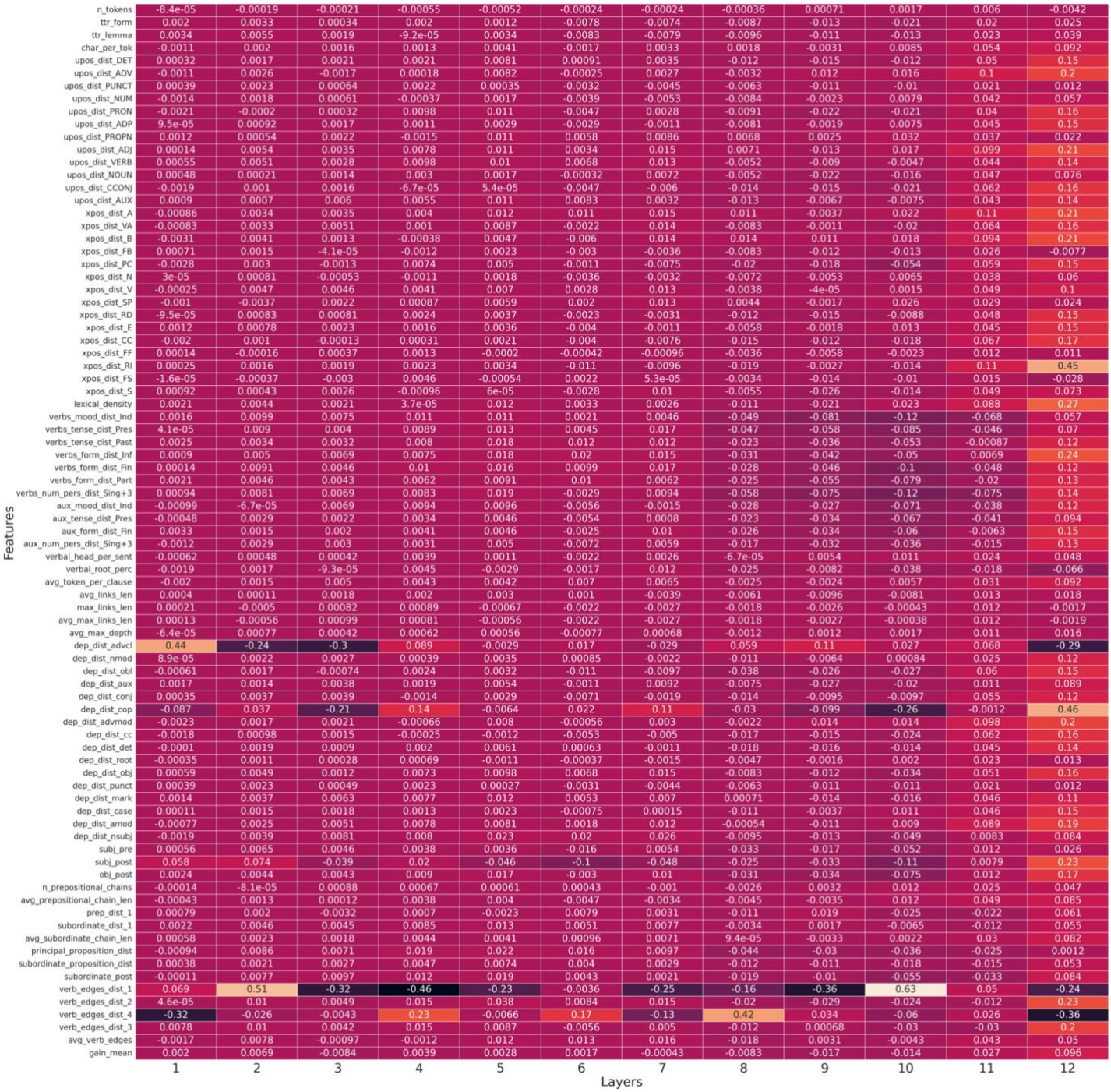


Figura 5.9. Heatmap con calcolo del gain tra modello con *fine tuning* sulla complessità e modello preaddestrato

Relativamente al confronto tra il modello con *fine tuning* sulla complessità e il modello preaddestrato, invece, le *feature* linguistiche con punteggio superiore nell'ultimo layer risultano: *dep_dist_cop*, *xpos_dist_RI*, *lexical_density*, *verbs_form_dist_Inf*, *subj_post*.

Prendendo in considerazione i ranking delle *feature* dell'ultimo layer, i due confronti (figura 5.8 e 5.9) risultano molto correlati tra loro, ottenendo un punteggio di 0.91. Ancora una volta, questo sottolinea la somiglianza tra i due compiti linguistici di accettabilità e complessità, in quanto dopo il *fine tuning* sui due task si sviluppano competenze simili per la risoluzione del problema che si vuole affrontare.

Andando, poi, a verificare la correlazione di Spearman tra questi due ranking di feature con le caratteristiche linguistiche più rilevanti[16] per il regressore lineare descritto nel capitolo 4.1, sono emersi punteggi vicini allo zero. In particolare, la correlazione tra il ranking di feature dell'ultimo layer del gain del modello con *fine tuning* sull'accettabilità e le caratteristiche linguistiche più rilevanti per il *regressore lineare* addestrato sullo stesso task è risultata essere 0.03. Nel caso, invece, della complessità si è ottenuto un punteggio di 0.05.
Non essendoci correlazione si deduce che il modo in cui viene immagazzinata conoscenza in un *regressore lineare* è molto diverso da quanto avviene in BERT.

Un'ulteriore analisi statistica interessante è stata svolta calcolando la correlazione tra i due ranking di feature del gain e le caratteristiche linguistiche correlate con i compiti di accettabilità e complessità (descritte nel capitolo 3.2 e riportate rispettivamente nelle tabelle 3.3 e 3.4). Mentre nel caso del confronto relativo all'accettabilità si è ottenuta una correlazione pressoché pari a zero (0.02), per quanto riguarda la complessità il valore è risultato essere -0.24. Questo significa che vi è una correlazione abbastanza significativa tra il *gain* delle *feature* del modello messo a punto sulla complessità e le caratteristiche linguistiche più correlate con la complessità.

[16] Riportate nelle tabelle 4.2 (per quanto riguarda il compito dell'accettabilità) e 4.3 (per il task della complessità).

6. Conclusioni

Il presente lavoro di tesi si è concentrato sulla valutazione dell'accettabilità e la complessità linguistica di una frase utilizzando un *Neural Language Model* (BERT). In particolare, è stato analizzato come le rappresentazioni interne alla rete neurale cambiano quando addestrata a predire la percezione di un parlante rispetto ai due fenomeni.

Partendo dalla constatazione che i due compiti di accettabilità e complessità sono negativamente correlati tra di loro (-0.49) e che più una frase è percepita come complessa, meno è accettabile, si è cercato di rispondere alle domande di ricerca iniziali, su (1) quali sono le caratteristiche linguistiche che rendono una frase più o meno accettabile o complessa, (2) quanto un Neural Language Model riesce a modellare giudizi umani di accettabilità e complessità linguistica, (3) come varia, nei diversi layer, la conoscenza di un NLM dopo il *fine tuning* sui due compiti linguistici.

In primo luogo, per indagare i fenomeni linguistici statisticamente più correlati con l'accettabilità e la complessità, per ciascuna frase della porzione del corpus di *AcCompl-it*, è stato creato un vettore di caratteristiche linguistiche (che vanno da caratteristiche di base fino a caratteristiche sintattiche più complesse). Prendendo in esame ciascuna dimensione dei vettori è stata calcolata la correlazione di Spearman tra le *feature* e i punteggi di accettabilità e complessità. La caratteristica maggiormente correlata con il punteggio di accettabilità è risultata *verb_num_dist_Sing+3*, con un punteggio di 0.32. Questo indica che i verbi alla terza persona singolare vengono percepiti come maggiormente accettabili. Inoltre, sempre considerando l'accettabilità, si è mostrata significativa la distribuzione delle preposizioni e dei modificatori avverbiali. Questo indica che all'aumentare della distribuzione di queste due caratteristiche, aumenta pure il grado di accettabilità di una frase. Tra le feature linguistiche che, inoltre, rendono una frase meno accettabile spicca il *numero di token* (con un punteggio di -0.28): infatti, più una frase è lunga, più è considerata come meno accettabile. È interessante notare che, come ci aspettavamo, tale caratteristica risulta, invece, essere quella maggiormente correlata in senso positivo (0.60) con il compito linguistico della complessità. Tra le *feature* più correlate in senso negativo,

contrariamente, troviamo la distribuzione della punteggiatura (-0.42). Dunque, questa *feature* fa percepire una frase come meno complessa.
Confrontando i *ranking* delle caratteristiche linguistiche maggiormente correlate con i due task, tramite una correlazione per ranghi di Spearman, si è ottenuto un punteggio di -0.37. Questo valore negativo dimostra, ancora una volta, quanto questi due compiti sono correlati e suggerisce che le feature che rendono la percezione della frase come più complessa, al contempo contribuiscono a renderla meno accettabile.

Per rispondere alla seconda domanda di ricerca e comprendere quanto un Neural Language Model sia in grado di modellare i fenomeni di accettabilità e complessità rispetto a un sistema che utilizza feature linguistiche esplicite, è stata valutata la capacità sia di un regressore lineare (un Support Vector Regression con kernel *linear*), che di BERT, nel predire i punteggi su una scala Likert da 1 a 7 di accettabilità e complessità, assegnati alle 672 frasi del dataset *AcCompl-it*. In entrambi i casi, è stato effettuato un 12 *k-fold cross validation* e, come metrica di valutazione, è stata calcolata la correlazione tra i valori predetti dai due sistemi e i valori reali.
Il regressore lineare (SVR) ha ottenuto un punteggio di 0.59 per la predizione dei valori di accettabilità e di 0.70 per quanto riguarda la complessità. BERT, invece, in entrambi i casi raggiunge score superiori: in particolare, ottiene 0.77 nel primo compito e 0.74 nel secondo. Questi risultati dimostrano che le feature linguistiche esplicite sono utili a risolvere i due compiti, infatti, i risultati del regressore lineare sono soddisfacenti. Tuttavia, anche senza l'utilizzo di queste, BERT è in grado di risolvere i task e individuare informazioni che permettano di ottenere punteggi migliori.
Successivamente, appurata l'alta correlazione tra i due compiti linguistici di accettabilità e complessità, sono stati elaborati modelli di *transfer learning*, che utilizzano la conoscenza precedentemente acquisita per la risoluzione di un problema, per affrontarne uno diverso. In particolare, sono stati messi a punto due modelli addestrandoli prima sul task della complessità e poi su quello dell'accettabilità (e viceversa). In questo modo le performance sono ulteriormente aumentate: per quanto riguarda il modello con *fine tuning* sulla complessità e sull'accettabilità si è ottenuto uno score di 0.85, mentre relativamente al modello con *fine tuning* sull'accettabilità e la complessità si è raggiunto un punteggio di 0.76. Questo risultato suggerisce che avere precedentemente acquisito conoscenza su un altro task (accettabilità o

complessità) rispetto a quello su cui si devono fare le predizioni finali è di grande aiuto per il modello, proprio perché i due compiti sono molto simili.

Per comprendere meglio quali *feature* linguistiche hanno guidato maggiormente il modello di regressione lineare (SVR) nell'assegnazione del punteggio di accettabilità e complessità, sono stati analizzati i pesi assegnati a ciascuna caratteristica per la risoluzione dei task. Per quanto riguarda l'accettabilità, le caratteristiche con maggior peso sono risultate essere quelle verbali (*dep_dist_aux*, *verbs_tense_dist_Fut*, *upos_dist_AUX*). Considerando che, già precedentemente, la caratteristica più correlata con questo compito linguistico era una *feature* verbale, si può dedurre che per la valutazione dell'accettabilità giocano un ruolo fondamentale i verbi.
Relativamente alla complessità, invece, è interessante notare che la caratteristica con peso maggiore risulta essere *n_tokens*, che era esattamente la *feature* che prima avevamo dimostrato essere più correlata con questo compito linguistico. Inoltre, tra le caratteristiche più importanti per il regressore lineare troviamo la distribuzione dei verbi al passato e delle subordinate. Pure queste due *feature* erano precedentemente risultate tra le più correlate con il compito della complessità.

Per valutare quanto il regressore lineare (SVR) riesca a individuare le caratteristiche maggiormente correlate con l'accettabilità e la complessità, è stata, poi, calcolata la correlazione per ranghi di Spearman, tra i ranking delle feature più correlate con i due compiti e quelle con maggior peso secondo il modello di regressione. Per entrambi i ranking di caratteristiche linguistiche si è ottenuto un punteggio di circa 0.3: questo significa che, il regressore lineare è in grado di individuare e sfruttare le *feature* maggiormente correlate con i due task.
Inoltre, confrontando le liste delle caratteristiche linguistiche con maggior peso prodotte dall'SVR per i due task, è emerso un punteggio di correlazione di -0.65. Questo risultato (come già dimostrato precedentemente comparando i ranking delle feature più correlate con i task di accettabilità e complessità) suggerisce, ancora una volta, che i due compiti sono molto correlati e più una feature contribuisce a rendere una frase complessa, meno la rende accettabile.

Infine, per rispondere all'ultima domanda di ricerca su come varia, nei diversi layer, la conoscenza di un NLM dopo il *fine tuning* sui due compiti linguistici, sono state condotte due ulteriori analisi.

In primo luogo, è stata calcolata la similarità cosenica tra le rappresentazioni estratte (layer per layer) dai diversi modelli elaborati, sia per valutare le differenze con il modello preaddestrato, sia per studiare l'impatto del processo di *transfer learning*. È stato interessante notare che le rappresentazioni ottenute dopo il *fine tuning* diretto sui due task di accettabilità e complessità linguistica sono simili fino al penultimo strato e, solo successivamente, avviene la specializzazione specifica per la risoluzione del compito. Questo, ancora una volta, è una conseguenza dell'alta correlazione dei diversi task linguistici, che quindi spingono i modelli a carpire informazioni simili per la loro esecuzione.

Analizzando la similarità cosenica tra le rappresentazioni estratte dai modelli di *transfer learning* e quelle prodotte dai modelli messi a punto sui singoli task, è emerso che vi è una maggiore similarità tra le reti che hanno l'accettabilità come compito linguistico finale. In ogni caso, le differenze nelle rappresentazioni, introdotte dall'aver addestrato prima il modello su un altro task, contengono informazioni utili, in quanto si riflettono nelle performance superiori ottenute sui due task con i modelli di *transfer learning*.

Come secondo esperimento per comprendere l'effetto del *fine tuning* sulle competenze linguistiche, sono stati eseguiti 82 probing task, ciascuno corrispondente a feature che catturano il lessico, la morfosintassi e la sintassi di una frase. Tali probing task sono stati effettuati usando le rappresentazioni contestuali di ciascun modello per ogni frase del dataset IUDT e predicendo il valore reale di ciascuna feature linguistica lungo i layer interni. In questo modo, sono state individuate variazioni nell'informazione linguistica codificata tra i diversi modelli.

Analizzando i modelli messi a punto sui diversi compiti linguistici, rispetto alle performance ottenute dal semplice modello di BERT preaddestrato, si verifica un miglioramento dei punteggi nello strato di output. Tale incremento non era affatto scontato perché nella maggior parte dei lavori che hanno indagato le competenze linguistiche nell'ultimo strato dopo il *fine tuning*, si era verificato un peggioramento delle performance, dovuto alla specializzazione sul task finale. Il fatto che, in questo lavoro, si sia riscontrato un comportamento opposto indica che i due compiti di

accettabilità e complessità necessitano di grande competenza linguistica per essere risolti.
In generale (ad esclusione dei gruppi *RawText* e *Vocabulary*), tutti i gruppi di feature tendono a peggiorare intorno al decimo/undicesimo layer per poi avere una netta ripresa al dodicesimo. Questa tendenza è significativa perché mostra come i modelli organizzino l'informazione necessaria per la risoluzione dei compiti finali.

Per approfondire come varia la competenza linguistica di ciascuna feature dopo il processo di fine tuning è stato, poi, calcolato il *gain*, ossia la percentuale di conoscenza acquisita. Più nello specifico, si è misurata la percentuale di guadagno (in ciascuna feature) dei modelli con fine tuning sui diversi compiti linguistici rispetto al semplice modello BERT preaddestrato. Sia confrontando il modello con fine tuning diretto sull'accettabilità che sulla complessità con il modello preaddestrato, le caratteristiche linguistiche relative alle parti del discorso ottengono il maggior guadagno percentuale della conoscenza (con un aumento superiore al 10%), le caratteristiche relative al predicato verbale, invece, risultano quelle con minor incremento (circa l'1-2%). Andando, poi, a considerare i gruppi di feature nel confronto tra i modelli di transfer learning e BERT preaddestrato, per quanto riguarda la rete con fine tuning sulla complessità e sull'accettabilità, tra le caratteristiche con maggior incremento troviamo quelle relative all'ordine delle parole all'interno della frase (come la posizione del soggetto/oggetto rispetto al verbo o della subordinata rispetto alla principale). Mentre le caratteristiche con minor guadagno risultano essere quelle che riguardano il vocabolario (ad esempio la *type token ratio*).

Calcolando, successivamente, il gain tra i modelli di transfer learning e i modelli messi a punto sui singoli task, mentre, nell'ultimo strato, nel modello di *transfer learning* che predice l'accettabilità, la maggior parte delle caratteristiche ottiene punteggi superiori allo zero, nella rete di *transfer learning* addestrata sulla complessità, si riscontrano punteggi negativi (con un minimo di -4.5% per il gruppo *VerbInflection* e un massimo di -1% per il gruppo *VerbPredicate*), che evidenziano una lieve perdita di alcune competenze linguistiche.

Dopodiché, come ulteriore analisi statistica, è stato estratto, dal layer di output, il ranking delle caratteristiche linguistiche con maggior gain (ottenuto confrontando i

modelli con fine tuning diretto sui singoli task e il modello preaddestrato) ed è stata calcolata la correlazione di Spearman. Così facendo, si è ottenuto un punteggio di 0.91. Ancora una volta, questo sottolinea la somiglianza tra i due compiti linguistici di accettabilità e complessità, in quanto dopo il fine tuning sui due task si sviluppano competenze simili per la risoluzione del problema che si vuole affrontare.

Confrontando, inoltre, i due ranking del gain con le feature più importanti per il regressore lineare, si è ottenuto un punteggio di correlazione di circa zero (sia per la complessità che per l'accettabilità). Non essendoci correlazione si deduce che il modo in cui viene organizzata la conoscenza in un regressore lineare è molto diverso da quanto avviene in BERT.

Concludendo, in questo studio sono state indagate le tre domande di ricerca iniziali, analizzando come le similarità e le differenze nei due compiti linguistici di accettabilità e complessità vengono trattate da un Neural Language Model e da un regressore lineare SVR.
Questo lavoro di tesi potrà essere esteso in futuro andando ad effettuare ulteriori esperimenti su task non linguisticamente motivati (come, ad esempio, la *sentiment analysis*) per comprendere se le stesse considerazioni si possono estendere a compiti di natura diversa. Inoltre, potrebbe essere interessante analizzare il meccanismo di attenzione di un NLM rispetto ai compiti linguistici di valutazione di accettabilità/complessità ed esplorare questi task su un'altra lingua come, ad esempio, l'inglese (per capire se i risultati emersi in questo lavoro prescindono dalla lingua scelta).

7. Bibliografia

Adi, Yossi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. *Fine-grained analysis of sentence embeddings using auxiliary prediction tasks*. arXiv preprint arXiv:1608.04207.

Baroni, Marco, Georgiana Dinu, Germán Kruszewski. 2014. *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors*. In "Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics", pp. 238–247.

Belinkov, Yonatan, Llu´ıs M`arquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, e James Glass. 2017. *Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks*. In "Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)", pages 1–10.

Boleda, Gemma. 2020. *Distributional Semantics and Linguistic Theory*. In "Annual Review of Linguistics". 6:213-234.

Bosco, Cristina, Simonetta Montemagni, e Maria Simi. 2013. *Converting Italian Treebanks: Towards an Italian Stanford Dependency Treebank*. In "Proceedings of the ACL Linguistic Annotation Workshop & Interoperability with Discourse".

Brown, Tom, B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei. 2020. *Language Models are Few-Shot Learners*.

Brunato, Dominique, Andrea Cimino, Felice Dell'Orletta, Simonetta Montemagni, Giulia Venturi. 2020. *Profiling-UD: a Tool for Linguistic Profiling of Texts*. In "Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)", 7145-7151.

Brunato, Dominique, Cristiano Chesi, Felice dell'Orletta, Simonetta Montemagni, Giulia Venturi, Roberto Zamparelli. 2020. *AcCompl-it @ EVALITA2020: Overview of the Acceptability & Complexity Evaluation Task for Italian.*

Brunato, Dominique, Lorenzo De Mattei, Felice Dell'Orletta, Benedetta Iavarone, e Giulia Venturi. 2018. *Is this sentence difficult? do you agree?* In "Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing", pages 2690–2699.

Chesi, Cristiano, Paolo Canal. 2019. *Person features and lexical restrictions in Italian clefts*. In “Frontiers in Psychology”, 10:2105.

Cignarella, Alessandra, Teresa, Cristina Bosco, e Paolo Rosso. 2019. *Presenting TWITTIRÒ-UD: An italian twitter treebank in universal dependencies*. In “Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SynaxFest 2019)”.

Clark, Kevin, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. *What does BERT look at? An analysis of BERT’s attention*. In “Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP”, pages 276–286, Florence, Italy, August. Association for Computational Linguistics.

Conneau, Alexis, Germ´an Kruszewski, Guillaume Lample, Lo¨ic Barrault, and Marco Baroni. 2018. *What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties*. In “Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)”, pages 2126–2136.

De Leeuw, Joshua R. 2015. *jspsych: A javascript library for creating behavioral experiments in a web browser*. In “Behavior research methods”, 47(1):1–12.

Delmonte, Rodolfo, Antonella Bristot, e Sara Tonelli. 2007. *VIT - Venice Italian Treebank: Syntactic and quantitative features*. In “Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories”.

Delmonte, Rodolfo. 2020. *Venses@AcCompl-it: Computing complexity vs acceptability with a constituent trigram model and semantics*. In “Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020)”, Online. CEUR.org, Valerio Basile, Danilo Croce, Maria Di Maro, e Lucia C. Passaro, editori.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Erk, Katrin & Padó, Sebastian. 2008. *A Structured Vector Space Model for Word Meaning in Context*. 897-906.

Garvin, Paul. 1962. *Computer participation in linguistic research*. In “Language”, 38/4): 385-389.

Goldberg, Yoav. 2019. *Assessing bert’s syntactic abilities*. arXiv preprint arXiv:1901.05287.

Graffi, Giorgio, Sergio Scalise. *Le lingue e il linguaggio. Introduzione alla linguistica*. Bologna, Il Mulino, 2002.

Greco, Matteo, Paolo Lorusso, Cristiano Chesi, and Andrea Moro. 2020. *Asymmetries in nominal copular sentences: Psycholinguistic evidence in favor of the raising analysis*. In “Lingua”, 245:102926

Harris, Zellig Sabbetai. 1954. *Distributional structure*. In “Word”, 10.2-3, pp. 146–162.

Jain, Sarthak, Byron C. Wallace. 2019. *Attention is not Explanation*. In “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), pages 3543–3556, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Jawahar, Ganesh, Benoît Sagot, Djam´e Seddah, Samuel Unicomb, Gerardo Iniguez, Márton Karsai, Yannick Léo, Márton Karsai, Carlos Sarraute, Éric Fleury, e altri. 2019. *What does Bert learn about the structure of language?* In “57th Annual Meeting of the Association for Computational Linguistics (ACL)”, Florence, Italy.

Landauer, Thomas, K., Susan Dumais. 1997. *A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge*. In “Psychological Review”. 104(2), 211–240.

Lenci, Alessandro. 2018. *Distributional models of word meaning*. In “Annual Review of Linguistics”. 4: 151-71.

Levy, Omer, Yoav Goldberg, Ido Dagan. 2015. *Improving Distributional Similarity with Lessons Learned from Word Embeddings*. In “Transactions of the Association for Computational Linguistics”, 3 211–225.

Lin, Yongjie, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT’s linguistic knowledge. In “Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP”, pages 241–253, Florence, Italy, August. Association for Computational Linguistics

Linzen, T., E. Dupoux, e Y. Goldberg. 2016. *Assessing the ability of LSTMs to learn syntax-sensitive dependencies*. In “Transactions of the Association for Computational Linguistics”, volume 4, pages 521–535.

Liu, Nelson F., Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. *Linguistic knowledge and transferability of contextual representations*. In “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), pages 1073–1094, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Mancini, Simona, Paolo Canal e Cristiano Chesi. 2018. *The acceptability of person and number agreement/disagreement in Italian: an experimental study*. In "Lingbuzz preprint": https://ling.auf.net/lingbuzz/005514.

Marvin, Rebecca, Tal Linzen. 2018. *Targeted syntactic evaluation of language models*. In "Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing", pages 1192–1202.

Miaschi, Alessio, Dominique Brunato, Felice Dell'Orletta, Giulia Venturi. 2020. *Linguistic Profiling of a Neural Language Model*.

Mikolov, Tomas, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. *Efficient estimation of word representations in vector space*. In "Proceedings of Workshop at ICLR 2013*"*, pp. 1– 12.

Miller, George, Walter Charles. 1991. *Contextual correlates of semantic similarity, Language and Cognitive Processes*. In "Language, Cognition and Neuroscience", 6:1, 1-28.

Miller, George. 1967. *Empirical methods in the study of semantics*: 572-573.

Moro, Andrea. 1997. *The raising of predicates: Predicative noun phrases and the theory of clause structure*, volume 80. Cambridge University Press.

Nivre, Joakim, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, e altri. 2016. *Universal dependencies v1: A multilingual treebank collection*. In "Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)", pages 1659–1666.

Perone, Christian S., Roberto Silveira, and Thomas S Paula. 2018. *Evaluation of sentence embeddings in downstream and linguistic probing tasks*. arXiv preprint arXiv:1806.06259.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettle-moyer. 2018. *Deep contextualized word representations*. In "Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies", Volume 1 (Long Papers), pages 2227–2237.

Pustejovsky, James, Batiukova Olga. 2019. *The Lexicon*. In "Cambridge University Press".

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. *Improving language understanding by generative pre-training*.

Rizzi, Luigi, Giorgio Graffi. *La sintassi generativo-trasformazionale*. Bologna, Il Mulino, 1979

Rocchietti, Guido, Flavia Achena, Giuseppe Marziano, Sara Salaris, Alessandro Lenci. 2021. *FANCY: A Diagnostc Data-Set for NLI Models*.

Sahlgren, Magnus, Alessandro Lenci. 2016. *The effects of Data Size and Frequency Range on Distributional Semantic Models*. In "Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing", 975-980.

Sanguinetti, Manuela e Cristina Bosco. 2015. *PartTUT: The turin university parallel treebank*. In "Harmonization and Development of Re- sources and Tools for Italian Natural Language Processing within the PARLI Project", page 51–69. Springer, Roberto Basili e altri editori.

Sanguinetti, Manuela, Cristina Bosco, Alberto Lavelli, Alessandro Mazzei, e Fabio Tamburini. 2018. *PoSTWITA-UD: an Italian Twitter Treebank in universal dependencies*. In "Proceedings of the Eleventh Language Resources and Evaluation Conference (LREC 2018)".

Sarti, Gabriele. 2020. *UmBERTo-MTSA @ AcCompl- it: Improving complexity and acceptability prediction with multi-task learning on self-supervised annotations*. In "Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020)", Online. CEUR.org, Valerio Basile, Danilo Croce, Maria Di Maro, e Lucia C. Passaro editori.

Shen, Dingan, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, Lawrence Carin. *Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms*. In "Proceedings of ACL", 440-450.

Tang, Gongbo, Rico Sennrich, and Joakim Nivre. 2018. *An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation*. In "Proceedings of the Third Conference on Machine Translation: Research Papers", pages 26–35, Belgium, Brussels, October. Association for Computational Linguistics.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick. 2019a. *BERT rediscovers the classical NLP pipeline*. In "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pages 4593–4601, Florence, Italy, July. Association for Computational Linguistics.

Tenney, Ian, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin, Van Durme, Samuel R Bowman, Dipanjan Das, e altri. 2019b. *What do you learn from context? probing for sentence structure in contextualized word representations*. arXiv preprint arXiv:1905.06316

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. *Attention is all you need*.

Warstadt, Alex, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun, Liu, Alicia Parrish, et al. 2019. *Investigating bert's knowledge of language: Five analysis methods with npis*. In "Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)", pages 2870–2880.

Wieting, John, Douwe Kiela. 2019. *No Training Required: Exploring Random Encoders for Sentence Classification*. In "Proceedings of ICLR 2019", 1-16.

Wieting, John, Mohit Bansal, Kevin Gimpel, Karen Livescu. 2016. *Towards Universal Paraphrastic Sentence Embeddings*. In "Proceedings of ICLR 2016", 1-19.

Williams, Adina, Nikita Nangia, and Samuel Bowman. 2018. *A broad-coverage challenge corpus for sen- tence understanding through inference*. In "Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies", Volume 1 (Long Papers), pages 1112–1122. Association for Computational Linguistics.

Zhu, Xunjie, Tingfeng Li, Gerard de Melo. 2018. *Exploring Semantic Properties of Sentence Embeddings*. In "Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)".

8. Sitografia

Alammar, Jay. *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. Consultato 11 febbraio 2022. https://jalammar.github.io/illustrated-bert/.

Alammar, Jay. *The Illustrated Transformer*. Consultato 11 febbraio 2022. https://jalammar.github.io/illustrated-transformer/.

Alammar, Jay. *Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)*. Consultato 11 febbraio 2022. https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/.

Bert base italian uncased. Consultato 11 febbraio 2022. https://huggingface.co/dbmdz/bert-base-italian-uncased.

Hugging Face. *Dbmdz BERT and ELECTRA models*. Consultato 14 marzo 2022. https://huggingface.co/dbmdz/bert-base-italian-uncased.

Italian Natural Language Processing Lab. *Profiling-UD*. Consultato 11 febbraio 2022. http://www.italianlp.it/demo/profiling-UD.

Pandas Dataframe correlation. Consultato 12 marzo 2022. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html.

ScikitLearn metrics pairwiseCosine Similarity. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html. Consultato 18 marzo 2022.

ScikitLearn model selection KFold. Consultato 12 marzo 2022. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html.

ScikitLearn preprocessing MinMaxScaler. Consultato 12 marzo 2022. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

ScikitLearn svm SVR. Consultato 12 marzo 2022. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

9. Appendice

9.1 Feature linguistiche estratte da Profiling UD

Di seguito l'elenco delle feature linguistiche estratte dalla piattaforma Profiling Ud, utilizzata per analizzare le frasi del dataset:

- Proprietà *raw* del testo:
  - [n_sentences]: numero totale di frasi;
  - [n_tokens]: numero totale di tokens;
  - [tokens_per_sent]: lunghezza media delle frasi in un documento, calcolata in termini del numero di parole per frase;
  - [char_per_tok]: numero medio di caratteri per parola (esclusa la punteggiatura);
- Varietà lessicale:
  - [ttr_lemma_chunks_100]: Type/Token Ratio (TTR) calcolato rispetto ai lemmi nei primi 100 token di un documento. Varia tra 1 (alta varietà lessicale) e 0 (bassa varietà lessicale);
  - [ttr_lemma_chunks_200]: Type/Token Ratio (TTR) calcolato rispetto ai lemmi nei primi 200 token di un documento. Varia tra 1 (alta varietà lessicale) e 0 (bassa varietà lessicale);
  - [ttr_form_chunks_100]: Type/Token Ratio (TTR) calcolato rispetto alle forme delle parole nei primi 100 token di un documento. Varia tra 1 (alta varietà lessicale) e 0 (bassa varietà lessicale);
  - [ttr_form_chunks_200]: Type/Token Ratio (TTR) calcolato rispetto alle forme delle parole nei primi 200 token di un documento. Varia tra 1 (alta varietà lessicale) e 0 (bassa varietà lessicale);
- Informazione morfosintattica:
  - [upos_dist_*]: distribuzione delle 17 categorie *core* delle parti del discorso definite nei tag POS delle Universal Dependencies (nel dettaglio al seguente link: https://universaldependencies.org/u/pos/index.html);

    - [lexical_density]: il valore corrisponde al rapporto tra le parole piene (nomi, nomi propri, verbi, aggettivi, avverbi) e il numero totale di parole in un documento;
- Morfologia flessiva:
    - [verbs_tense_dist_*]: distribuzione dei verbi in accordo al loro tempo: https://universaldependencies.org/u/feat/Tense.html;
    - [verbs_mood_dist_*]: distribuzione dei verbi in accordo al loro modo: https://universaldependencies.org/u/feat/Mood.html;
    - [verbs_form_dist_*]: distribuzione dei verbi in accordo alla loro forma https://universaldependencies.org/u/feat/VerbForm.html;
    - [verbs_gender_dist_*]: distribuzione dei verbi secondo il genere delle forme al participio, per le lingue che hanno questa caratteristica: https://universaldependencies.org/u/feat/Gender.html;
    - [verbs_num_pers_dist_*]: distribuzione dei verbi secondo il loro numero e la loro persona: https://universaldependencies.org/u/feat/Person.html;
- Feature sintattiche:
    - Struttura del predicato verbale:
        - [verbal_head_per_sent]: distribuzione media delle teste verbali nel documento, sul totale delle teste;
        - [verbal_root_perc]: distribuzione media delle radici date da un lemma etichettato come verbo, sul totale delle radici della frase;
        - [avg_verb_edges]: la valenza verbale, calcolata come il numero medio di legami di dipendenza istanziati (che coprono sia gli argomenti che i modificatori) che condividono la stessa testa verbale, escludendo la punteggiatura e gli ausiliari che hanno il ruolo sintattico di copula secondo lo schema UD;
        - [verb_edges_dist_*]: distribuzione dei verbi per classe di valenza (per esempio verbi con valenza 1, 2, ...);
    - Strutture ad albero parsate globalmente o localmente:
        - [avg_max_depth]: media delle profondità massime dell'albero estratte da ogni frase di un documento. La profondità massima è calcolata come il percorso più lungo (in termini di

collegamenti di dipendenza che si verificano) dalla radice dell'albero delle dipendenze a qualche foglia;

- [avg_token_per_clause]: lunghezza media delle clausole, calcolata in termini di numero medio di token per clausola, dove una clausola è definita come il rapporto tra il numero di token in una frase e il numero di testa verbale o copulare;
- [avg_links_len]: numero medio di parole che occorrono linearmente tra ogni testa sintattica e la sua dipendenza (escluse le dipendenze di punteggiatura);
- [avg_max_links_len]: media dei collegamenti di dipendenza più lunghi estratti da ogni frase di un documento;
- [max_links_len]: il valore del collegamento di dipendenza più lungo nel documento, calcolato in numero di token;
- [avg_prepositional_chain_len]: valore medio delle 'catene' preposizionali estratte per tutte le frasi del documento. Una catena preposizionale è calcolata come il numero di complementi preposizionali incassati dipendenti da un nome.;
- [n_prepositional_chains]: numero totale di "catene" preposizionali estratte per tutte le frasi del documento;
- [prep_dist_*]: distribuzione delle catene preposizionali per profondità;

- Ordine degli elementi:
  - [obj_pre]: distribuzione degli oggetti che precedono il verbo;
  - [obj_post]: distribuzione degli oggetti che seguono il verbo;
  - [subj_pre]: distribuzione dei soggetti che precedono il verbo;
  - [subj_post]: percentuale di oggetti che seguono il verbo;
- Relazioni sintattiche:
  - [dep_dist_*]: distribuzione media delle 37 relazioni sintattiche universali usate in UD: (https://universaldependencies.org/u/dep/index.html);
- Uso delle subordinate:
  - [principal_proposition_dist]: distribuzione delle proposizioni principali;

- [subordinate_proposition_dist]: distribuzione delle proposizioni subordinate, come definito nello schema UD: https://universaldependencies.org/u/overview/complex-syntax.html#subordination;
- [subordinate_post]: distribuzione delle proposizioni subordinate che seguono la principale;
- [subordinate_pre]: distribuzione delle proposizioni subordinate che precedono la principale;
- [avg_subordinate_chain_len]: lunghezza media delle catene subordinate, dove una "catena" subordinata è calcolata come il numero di clausole subordinate incastrate su una prima clausola subordinata;
- [subordinate_dist_*]: distribuzione delle subordinate per profondità;

10. Ringraziamenti

Nonostante l'emergenza medico-sanitaria degli ultimi due anni, che non mi ha permesso di vivere pienamente l'ambiente universitario, sono comunque contenta di aver incontrato professori e colleghi sempre pronti al confronto, che hanno saputo alleggerire il clima esterno, rendendo il percorso piacevole e sfidante.

In particolare, desidero ringraziare il professor Felice Dell'Orletta, che mi ha seguita durante la parte finale di questo percorso universitario, trasmettendomi importanti insegnamenti, che porterò con me nella futura carriera lavorativa.

Una menzione speciale, inoltre, va a Ludovica, Stefano, Lorenzo, Angelica, Aldo e Chiara Z., colleghi che, seppur a distanza, ho sentito vicini a me più che mai.

Un ringraziamento affettuoso va ai miei genitori che sono sempre stati al mio fianco, supportandomi in ogni mia scelta e dandomi la forza per affrontare tutte le sfide in questo percorso di crescita.

Sono grata anche ai miei zii e mia cugina, per il loro sostegno e per aver gioito insieme a me in ogni piccolo traguardo, mostrandomi sempre la loro fiducia in me.

Grazie a Davide, che è in assoluto la persona che più mi è stata accanto in questo periodo di tesi, offrendomi sempre un abbraccio nei momenti di difficoltà e credendo in me sin dal principio. Grazie a lui sono nati molteplici punti di vista interessanti, per me grande fonte di ispirazione.

Ringrazio le mie migliori amiche, Elvira e Irene, ormai presenti dagli anni del liceo. Grazie per aver reso più leggeri questi anni, offrendomi sempre un motivo di svago.

Dedico, infine, questa tesi a mio cugino Alessandro, che purtroppo non potrà festeggiare con me in questa giornata. Nonostante non possa esserci fisicamente, so che sarebbe orgoglioso del mio lavoro e mi applaudirebbe seduto in prima fila. Caro Alessandro, sappi che ti sento sempre con me.