Progetto Linguistica Computazionale II

Assessing DIScourse COherence in Italian TEXts, EVALITA 2023

Giulio Leonardi

A.A. 2023/24

UNIVERSITÀ DI PISA

Contents

1 Introduzione

In questo report sarà presentato lo sviluppo di diversi metodi di classificazione, per risolvere il task di *Assessing DIScourse COherence in Italian TEXts*, EVALITA 2023. In particolare, sarà oggetto di studio il sub-task di Last sentence classification: dati un paragrafo formato da 3-4 frasi (prompt), e una frase (target), il modello deve predire se quest'ultima segue il prompt nel testo origninale, quindi la coerenza di target e prompt concatenati.

Per i record della classe negativa, i target possono essere o delle frasi di un documento diverso dal prompt (20% dei casi), oppure la decima frase successiva dello stesso documento rispetto al prompt (80% dei casi). Questo rende il task significativamente più complesso, richiedendo al classificatore una maggiore capacità di comprensione della struttura linguistica.

Il dataset che sarà utilizzato è quello formato dalle trascrizioni italiane parte del Multilingual TEDx corpus. Quindi si tratta di testi piuttosto eterogenei, che spaziano dalla lingua standard a quella parlata, su diversi generi. Queste caratteristiche dei dati aumenteranno ancor di più la difficoltà per i modelli nel classificare la coerenza delle frasi.

Il training set è composto da 8000 record, mentre il test set da 800, in entrambi i casi le classi sono perfettamente bilanciate.

I classificatori testati sono stati divisi in quattro famiglie: basati su feature linguistiche, basati su n-grammi, basati su word embedding *Word2Vec*, basati su un LLM. Per ogni capitolo quindi si tratteranno gli esperimenti svolti in fase di validazione, e i modelli ritenuti più interessanti saranno testati sul test set.

2 Caratteristiche Linguistiche

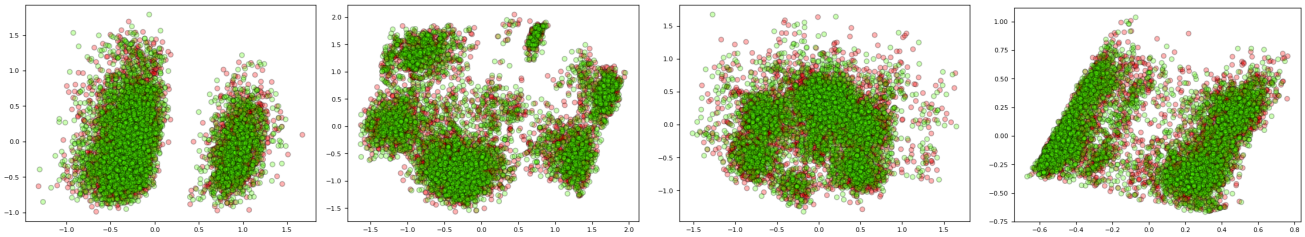
La prima rappresentazione dei dati sperimentata per la risoluzione del task è stata quella delle feature linguistiche. Da ogni testo sono state estratte 142 feature linguistiche, attraverso il tool Profiling-UD, allora sono stati formati quattro diversi dataset, distinti per il modo in cui trattano il prompt e il target:

- **Dataset 1**: prompt e target considerati come unico testo, quindi con un unico profiling.
- **Dataset 2**: prompt e target analizzati come testi separati, poi concatenando i due vettori di profiling.
- **Dataset 3**: prompt e target analizzati come testi separati, utilizzando come vettore rappresentativo la differenza assoluta dei due vettori di profiling.
- **Dataset 4**: prompt e target analizzati come testi separati, utilizzando la cosine similarity applicata a dei gruppi di feature del profiling.

Per spiegare più chiaramente il Dataset 4, quello che è stato fatto è suddividere le 142 feature del profiling in diversi gruppi seguendo il raggruppamento interno a Profiling-UD: varietà lessicale, informazioni morfosintattiche generali, morfologia flessiva, struttura dei predicati verbali, struttura degli alberi globali e locali, ordine degli elementi, relazioni sintattiche e fenomeni di subordinazione. A questo punto, per ognuno di questi gruppi, sono state selezionate

le relative feature per il prompt e per il target, ed è stata calcolata la cosine similarity, che è utilizzata come nuova feature rappresentativa del gruppo. Il vettore rappresentativo risultante è quindi composto da otto feature, una per ogni gruppo. L'idea alla base di questo approccio è la supposizione che, il modo migliore per sfruttare le feature linguistiche è quello di paragonare la similarità tra il prompt e il target, ma, facendolo su sutte le feature, si potrebbe incorrere nel fenomeno della *Curse Of Dimensionality*, rendendo difficile al classificatore il compito di trovare il limite decisionale per separare le classi. Invece, in uno spazio a meno dimensioni, le istanze dovrebbero tendere ad essere più separabili.

Successivamente, per valutare l'impatto spaziale di queste quattro diverse trasformazioni, ai dataset è stata applicata una *Principal Component Analisys* (solo per questa fase di visualizzazione, non in modo definitivo), così, sono stati realizzati dei grafici che mostrano le prime due dimensioni che catturano la massima varianza dei dati.



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3 (d) Dataset 4

Figure 1: Distribuzione dei record del training set in uno spazio bidimensionale ottenuto con la PCA; i rossi sono i negativi, i verdi i positivi.

Dalla figura 1 possiamo notare che i diversi metodi di combinazione del prompt e del target definiscono un diverso spazio delle istanze, questo può significare che è sensato il provare questi diversi approcci, perché non equivalenti tra loro. Un'altra osservazione che possiamo fare dal grafico, è che in tutte le rappresentazioni, le istanze positive e negative non sembrino facilmente separabili, questo potrebbe confermarci la difficoltà intrinseca di questo task (anche se comunque stiamo guardando solo le prime due componenti della PCA).

Per la classificazione è stata usata una Support Vector Machine lineare, in modo da rendere facilmente interpretabili i modelli. Di seguito è riportata l'accuracy in validazione per ogni dataset.

Dataset 1	**Dataset 2**	**Dataset 3**	**Dataset 4**
0.52	0.51	0.51	0.54

Table 1: Accuracy della 5-fold crossvalidation per ogni dataset

Come possiamo notare, i risultati sono solo leggermente migliori rispetto alla baseline (0.5), il modello migliore è quello addestrato sul Dataset 4, che sembra avere una minima capacità di distinzione delle classi. Per i modelli addestrati sui primi tre dataset, sembra che i classificatori stiano andando a caso. Per approfondire le prestazioni dei modelli, è il caso osservare i risultati sul test set.

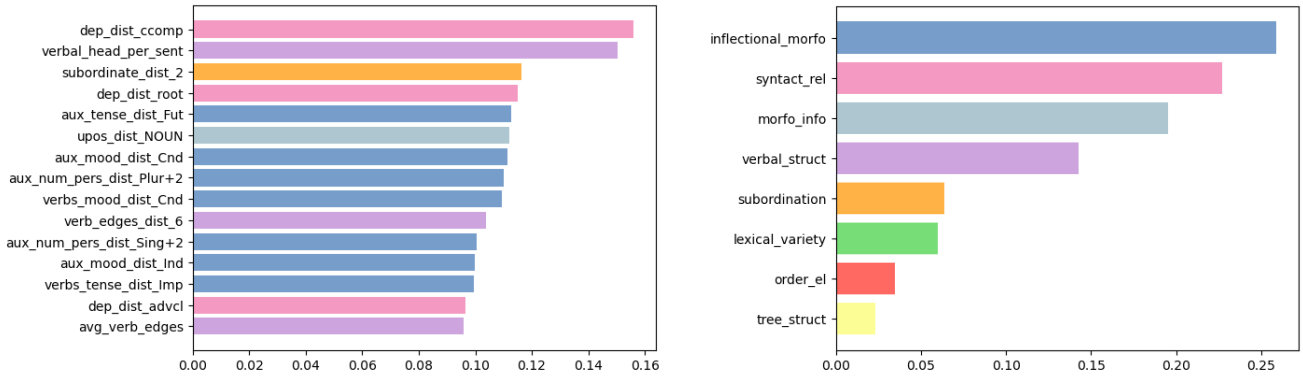
-	**Dataset 1**	**Dataset 2**	**Dataset 3**	**Dataset 4**
Accuracy	0.49	0.50	0.53	0.55
F1-score	0.49	0.50	0.53	0.54

Table 2: Prestazioni sul test set della classificazione basata sulle feature linguistiche

Dai risultati in tabella 2, osserviamo che i classificatori addestrati sui dataset 1 e 2 non hanno dimostrato alcuna capacità predittiva. Invece, la costruzione dei dataset 3 e 4, entrambi basati sull'assunzione che se il prompt e il target sono incoerenti allora hanno delle caratteristiche linguistiche leggermente diverse tra loro, ha condotto a delle rappresentazioni dei dati migliori, che hanno permesso ai classificatori delle prestazioni leggermente superiori.

Risulta comunque evidente che le feature linguistiche non possano bastare per questo task; in particolare nell'80% degli esempi della classe negativa, il target appartiene comunque allo stesso testo del prompt. Quindi è evidente che l'assunzione precedentemente presentata, sulla quale si basano i dataset 3 e 4, non sia rispettata da gran parte dei dati.

Infine, abbiamo analizzato i coefficienti della SVM lineare, per valutare le feature importance di ogni classificatore addestrato. Sono stati però analizzati unicamente i coefficienti dei classificatori addestrati sui dataset 3 e 4, perché sono gli unici classificatori che sembrano aver effettivamente imparato qualcosa dai dati.



(a) Dataset 3

(b) Dataset 4

Figure 2: Feature importance dei classificatori, i colori rappresentano i gruppi di feature linguistiche del Dataset 4.

Osservando i grafici in figura 2, le classifiche delle feature più importanti per i due classificatori sembrano coerenti, infatti le feature più importanti per il modello addestrato sul Dataset 3, fanno parte dei gruppi più importanti per il classificatore del Dataset 4. Notiamo quindi che le caratteristiche di morfologia flessiva e relazioni sintattiche sono molto importanti per decretare la coerenza tra prompt e target.

3 N-grammi

Il secondo approccio di rappresentazione utilizzato è quello degli n-grammi: ogni documento (prompt e target concatenati) è rappresentato dalla frequenza dei suoi n-grammi, normalizzata rispetto alla lunghezza del documento. In questo caso quindi i testi sono rappresentati dal proprio lessico. Sono stati formati quindi quattro dataset, distinti dal tipo di n-grammi utilizzati:

1. **Forma**: da 1-grammi a 5-grammi di forme.
2. **Lemma**: da 1-grammi a 5-grammi di lemmi.
3. **POS**: da 1-grammi a 5-grammi di Part of Speech.
4. **Carattere**: suffissi e prefissi delle parole, lunghi da 1 a 5 caratteri.

Per ridurre la dimensionalità dei dataset, è stato applicato un filtro, eliminando gli n-grammi con meno di 5 occorrenze nel training set. I dataset filtrati hanno rispettivamente 45829, 47293, 25641 e 18544 feature.

Per la classificazione, anche i questo caso sono state addestrate delle Support Vector Machine lineari.

Forma	**Lemma**	**POS**	**Carattere**
0.48	0.47	0.49	0.50

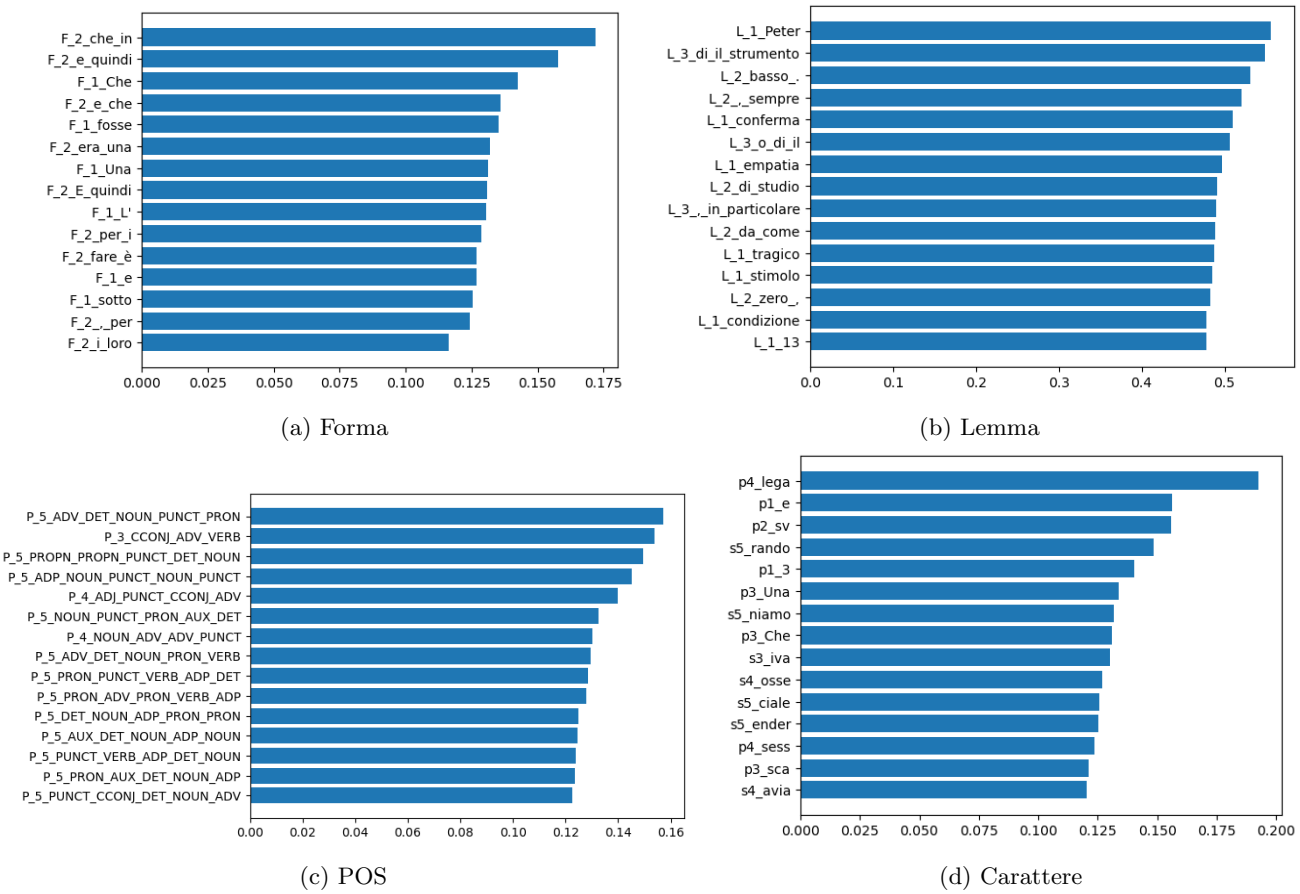
Table 3: Accuracy della 5-fold crossvalidation per ogni dataset

I risultati della 5-folds crossvalidation sul training set non sono promettenti, nessun modello va oltre il 50% di accuracy, sembra quindi che questo tipo di rappresentazione non sia adatta per risolvere il task.

-	**Forma**	**Lemma**	**POS**	**Carattere**
Accuracy	0.49	0.51	0.50	0.48
F1-score	0.49	0.50	0.50	0.48

Table 4: Prestazioni sul test set della classificazione basata sugli n-grammi

Osservando i risultati sul test set, possiamo confermare che i modelli non hanno alcuna capacità predittiva della coerenza dei testi; questo approccio si è quindi rivelato piuttosto fallimentare per la risoluzione del task. Come si può notare anche dal capitolo precedente, risolvere questo task senza le capacità di modellazione della lingua di un LLM, risulta piuttosto complesso. Per ottenere qualcosa di positivo è fondamentale un approccio di feature engineering nel combinare il prompt e il target. In questo caso, gli n-grammi definiscono dei vettori sparsi ad altissima dimensionalità, diventa quindi molto complesso trattarli e combinarli in modo ingegnoso rispetto agli altri approcci presentati in questo report.



(a) Forma

(b) Lemma

(c) POS

(d) Carattere

Figure 3: Feature importance dei classificatori basati su n-grammi

Per completezza, nella figura 3 sono riportate le feature con maggiore importanza per ogni classificatore; anche se, viste le prestazioni fallimentari della predizione, non possiamo affidarci molto all'importanza di queste feature. Infatti, non sembrano esserci particolari pattern comuni tra i diversi modelli, se non per la parola "Che" e l'articolo "Una", che sono presenti in classifica sia per il modello basato su n-grammi di forma, che su quello basato su n-grammi di carattere.

4 Word Embedding

In questa sezione sono presentati gli esperimenti di classificazione svolti a partire dal word embedding del lessico dei testi, attraverso l'algoritmo *Word2Vec*. Grazie a quest'algoritmo, a partire dalle parole, possiamo ottenere dei vettori numerici che catturano tutti i loro significati; inoltre, i rapporti geometrici tra i vettori rispecchiano i rapporti tra le parole della lingua. Quindi, con i word embedding, possiamo ragionare in termini di distanze e similarità tra i vettori delle parole.

In questo progetto, è stato utilizzato un insieme di word embedding pre-addestrato, in particolare la rappresentazione a 128 dimensioni ricavata dal corpus *itWaC*, che contiene testi italiani ricavati dal web.

Sono stati definiti tre diversi dataset, distinti tra loro per il modo in cui combinano il prompt e il target per ottenere un vettore unico rappresentativo per ogni testo.

- Dataset **Mean**: dopo aver estratto gli embedding di prompt e target, vengono tutti raggruppati e viene usata la media tra tutti come vettore rappresentativo.
- Dataset **Diff**: dopo aver estratto gli embedding di prompt e target, viene calcolato il vettore mediana del prompt, e il vettore mediana del target; la differenza assoluta tra i due è il vettore rappresentativo del documento.
- Dataset **Min_Dist**: dopo aver estratto gli embedding di prompt e target, calcolo la distanza euclidea tra tutte le coppie di embedding del prompt e del target, e considero solo la coppia di embedding con la distanza minima, e uso la differenza assoluta tra i due come vettore rappresentativo.

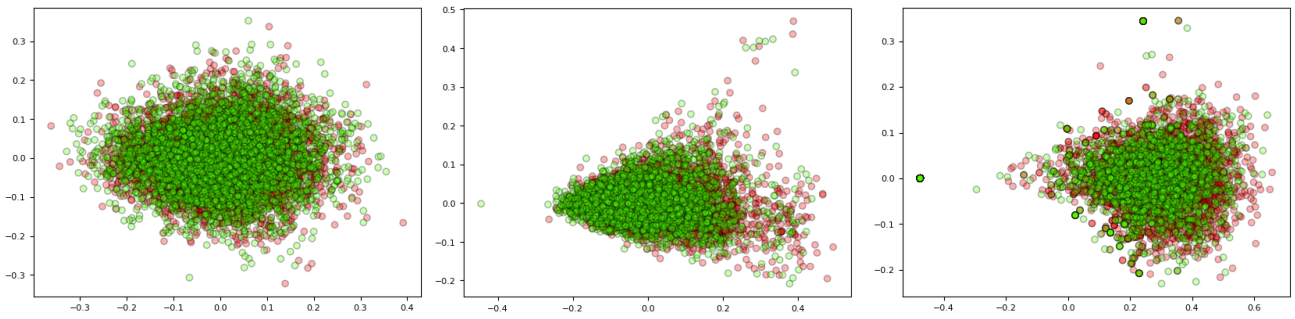
Inoltre, per la costruzione dei tre dataset, sono stati considerati sono i vettori di embedding di certe parti del discorso:

- Dataset **Mean**: sostantivi, nomi propri e verbi (non ausiliari).
- Dataset **Diff**: sostantivi, nomi propri, verbi (compresi gli ausiliari) e aggettivi.
- Dataset **Min_Dist**: sostantivi, nomi propri, verbi (non ausiliari), e aggettivi.

La scelta delle categorie grammaticali da includere per ogni dataset è dovuta innanzitutto all'intuizione: le parole contenuto sono quelle che veicolano gran parte della semantica della frase, sono quindi importati per valutare la coerenza, infatti, sostantivi e verbi sono stati utilzzati per ogni dataset. Successivamente, sono stati fatti dei test di prestazioni in 5-fold crossvalidation per valutare altre possibili categorie da aggiungere, quindi attraverso una forward selection, valutando per ogni aggiunta se ci fosse un miglioramento dei risultati. In questo modo si è arrivati alla scelta finale, che dovrebbe essere comunque una soluzione sub-ottimale.

Intuitivamente poteva essere interessante includere i pronomi, che potrebbero essere utili per distinguere la coerenza della frase, ma l'aggiunta di essi, abbassava notevolmente i risultati in validazione.

Anche in questo caso, per provare a visualizzare lo spazio delle diverse rappresentazioni, è stata applicata una PCA, mantenendo solo le prime due componenti.



(a) Dataset Mean (b) Dataset Diff (c) Dataset Min_Dist

Figure 4: Distribuzione dei record del training set in uno spazio bidimensionale ottenuto con la PCA; i rossi sono i negativi, i verdi i positivi.

Osservando le diverse distribuzioni delle istanze nei grafici in figura 4, possiamo sviluppare diverse osservazioni: le istanze positive e negative nel dataset Mean sembrano distribuite in modo abbastanza sparso, non sembra quindi una rappresentazione promettente per la discriminazione delle classi. La distribuzione del dataset Diff sembra invece più interessante, le istanze negative infatti sembrano avere una leggera tendenza verso la zona a destra della massa di punti. Infine, la rappresentazione più promettente in assoluto, è quella del dataset Min_Dist, infatti i record positivi sembrano raggrupparsi maggiormente nel centro del cluster, mentre i record negativi mostrano una tendenza a posizionarsi nella zona più esterna; questo potrebbe indicarci uno spazio dove le istanze sono più facili da separare per un classificatore.

Anche in questo caso l'algoritmo di classificazione utilizzato è una Support Vector Machine lineare, di seguito sono riportati i risultati in validazione e sul test set.

Mean	Diff	Min_Dist
0.51	0.56	0.63

Table 5: Accuracy della 5-fold crossvalidation per ogni dataset

-	Mean	Diff	Min_Dist
Accuracy	0.47	0.58	0.63
F1-score	0.47	0.58	0.62

Table 6: Prestazioni sul test set della classificazione basata sul Word Embedding

Le osservazioni tratte dalle distribuzioni in due dimensioni (figura 4), sembrano essere confermate dai risultati della classificazione. La rappresentazione del dataset Mean non permette una discriminazione delle classi, i risultati sono quindi pessimi. Invece, il dataset Diff, riporta un buon 58% di accuracy, e il dataset Min_Dist raggiunge un ottimo 63%.

Questi risultati confermano l'importanza della feature engineering per risolvere questo task senza un LLM, infatti solo gli approcci più raffinati di combinazione e aggregazione dei vettori di word embedding hanno portato dei buoni risultati.

5 Bert Fine-Tuning

L'ultimo approccio utilizzato per provare a risolvere il task di classificazione della coerenza tra le frasi, è il fine-tunig di un Large Language Model. In questo caso, è stato utilizzato il modello BERT base italiano del *MDZ Digital Library team (dbmdz)*.

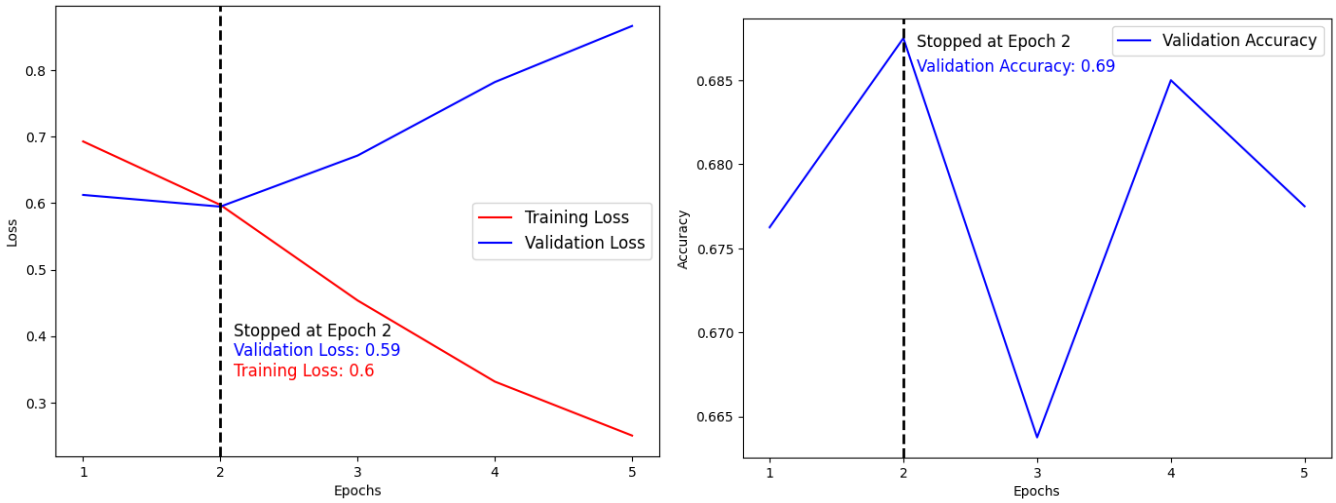
Quindi, per ogni istanza, il prompt e il target sono stati considerati come un unico testo, e sono stati tokenizzati attraverso il tokenizzatore associato al modello. Inoltre, dal training set, è stato ricavato un set di validazione di 800 istanze, che è servito per valutare l'addestramento del modello.

Sono stati effettuati tre esperimenti diversi:

- **Sequence Classification**: fine-tuning per 5 epoche del modello con la testa di classificazione per la classificazione di sequenze.
- **Next Sentence Prediction**: fine-tuning per 5 epoche del modello con la testa di classificazione adatta per il task di next sentence prediction.
- **Custom Classification Head**: fine-tuning di BERT con una testa di classificazione personalizzata.

5.1 Addestramento del Sequence Classification Model

Il modello è stato addestrato per cinque epoche, con un learning rate di 2e-5, la scelta di un learning rate basso serve ovviamente a evitare aggiornamenti troppo corposi dei pesi, che porterebbero a una perdita delle capacità di modellazione del linguaggio di BERT. Di seguito le curve della loss durante l'addestramento.



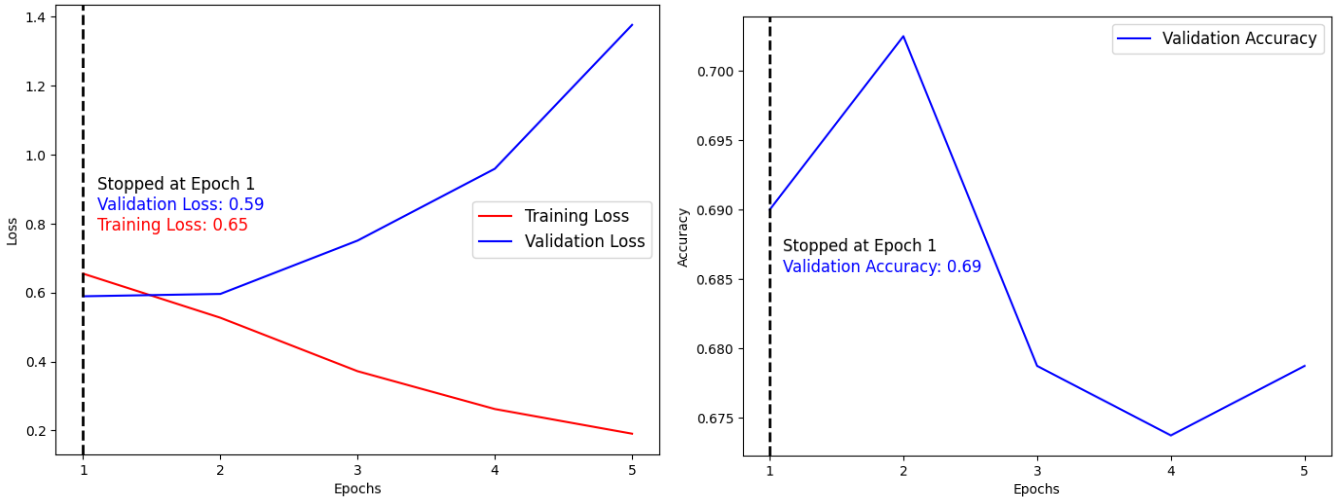
(a) Loss (b) Validation Accuracy

Figure 5: Andamento delle statistiche durante l'addestramento del modello di Sequence Classification.

Come si può notare dai grafici in figura 5, le migliori prestazioni sul validation set sono state raggiunte già alla seconda epoca, nelle successive il modello inizia a overfittare; infatti si può vedere come la loss sul training set continui a minimizzarsi, mentre la loss sul validation set aumenta, quindi il classificatore inizia a perdere capacità di generalizzazione. L'algoritmo di addestramento è stato impostato per mantenere i pesi migliori, ovvero quelli che hanno registrato la validation loss più bassa; quindi, il modello finale che sarà utilizzato sul test set è quello della seconda epoca.

5.2 Addestramento del Next Sentence Prediction Model

Anche in questo caso il modello è stato addestrato per cinque epoche con un learning rate di 2e-5. Di seguito le curve della loss durante l'addestramento.



(a) Loss (b) Validation Accuracy

Figure 6: Andamento delle statistiche durante l'addestramento del modello di Next Sentence Prediction.

Anche in questo caso, già dopo le prime due epoche, il modello inizia a overfittare in modo evidente. In questo caso il modello migliore è stato quello ottenuto alla fine della prima epoca, e le sue prestazioni sul validation set (69% di accuracy) sembrano in linea con il modello di Sequence Classification.

5.3 Addestramento del Custom Classification Head Model

L'idea alla base di questo esperimento, è che il layer lineare usato dal modello base per la classificazione, possa non riuscire a gestire la complessità del task, quindi, il tentativo è stato quello di sostituirlo con un Multilayer Perceptron.

Il classificatore utilizzato è una rete neurale densamente connessa, con quattro hidden layer, con rispettivamente con 128, 256, 512 e 128 neuroni, tutti con funzione di attivazione ReLU. Il layer di output invece, è composto da un solo neurone, con attivazione Sigmoid. Inoltre, è stata aggiunta una connessione residua tra l'output del primo hidden layer, e il layer di output. Per evitare l'overfitting, dopo ogni hidden layer è stato applicato il dropout, con probabilità di 0.5.

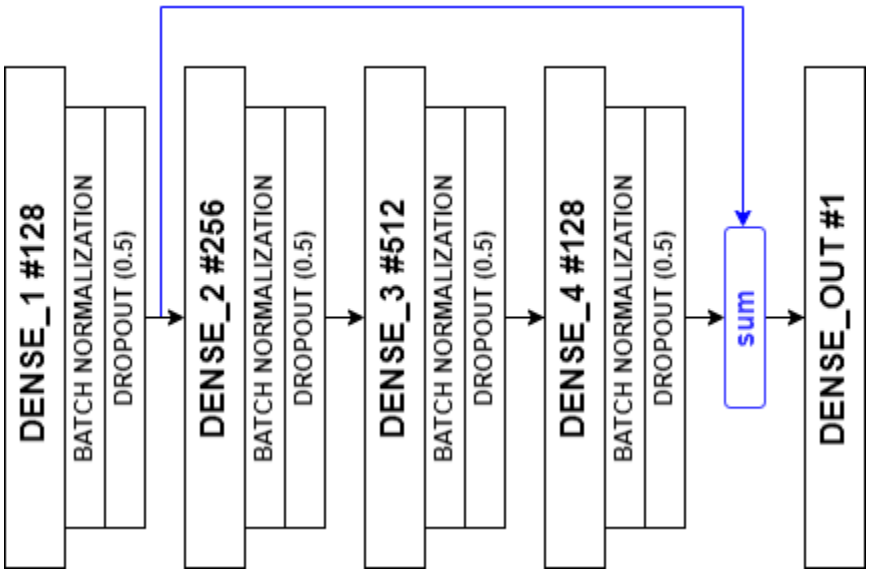
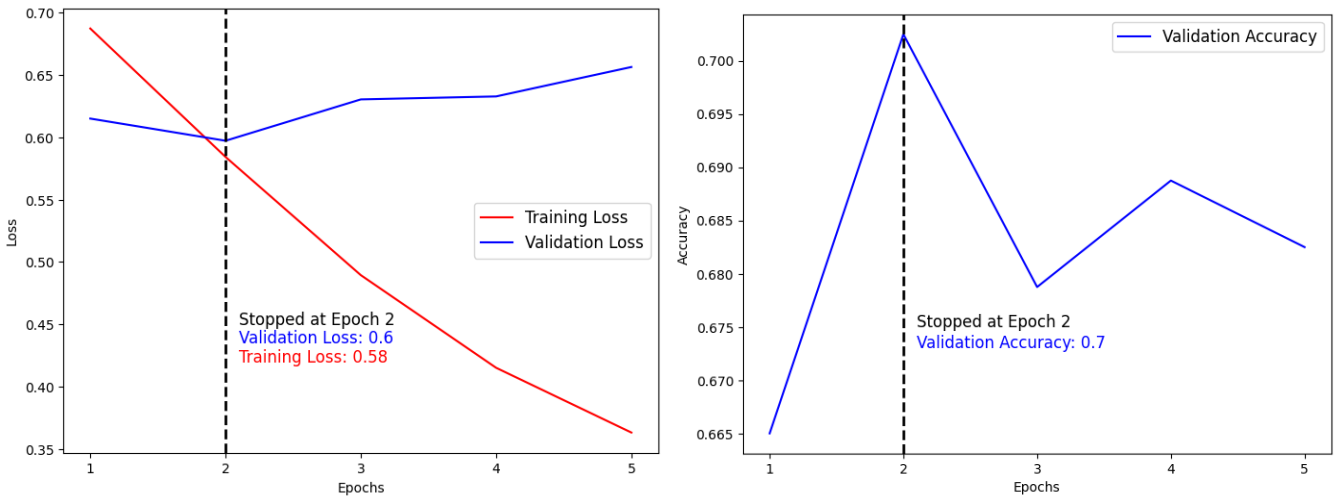


Figure 7: Architettura del classificatore

Il classificatore non può essere addestrato direttamente insieme a BERT, infatti, avendo i pesi appena inizializzati, il classificatore ha bisogno di un addestramento con un learning rate più alto e su più epoche. Quindi, il primo passo è stato quello di usare BERT senza testa di classificazione per estrarre le feature dal training set; successivamente, con questi dati è stato svolto un pre-addestramento del classificatore. Quindi è come se stessimo congelando l'addestramento dei pesi del transformer, addestrando solamente il classificatore.

Il pre-addestramento è stato svolto con l'algoritmo di ottimizzazione ADAM, con un learning rate di 0.001, un weight decay di 0.0001; la loss function utilizzata è stata la Binary Cross-Entropy. L'addestramento è stato fermato dopo 35 epoche, con una accuracy in validazione del 58%.

Successivamente, l'intero modello è stato addestrato insieme per cinque epoche, con un learning rate di 2e-5.



(a) Loss

(b) Validation Accuracy

Figure 8: Andamento delle statistiche durante l'addestramento del modello con testa di classificazione personalizzata.

Anche in questo caso si può notare che già dopo la seconda epoca il modello inizia a overfittare. In generale, l'addestramento sembra il linea con gli altri due modelli.

5.4 Analisi dei risultati

-	**Sequence Clf.**	**Next Sentence Clf.**	**Custom Clf.**
Accuracy	0.66	0.65	0.68
F1-score	0.66	0.65	0.67

Table 7: Prestazioni sul test set della classificazione basata sul fine tuning di BERT

Come si può osservare dalla tabella 7, i risultati dei tre modelli sono abbastanza allineati tra loro; il modello con le migliori prestazioni è quello con la testa di classificazione personalizzata, ma uno scarto così piccolo potrebbe anche essere dovuto a una inizializzazione fortunata dei pesi. Quindi, come prevedibile, sembra che gran parte della capacità predittiva dipenda dal modello del linguaggio, e non dalla testa di classificazione.

6 Conclusioni

In questo progetto, è stato affrontato il task di predizione della coerenza tra un paragrafo prompt, e una frase target. Il task è reso complesso dal fatto che nell'80% dei record della classe negativa, il target sia comunque una frase dello stesso documento del prompt. Quindi, il predittore non può semplicemente basarsi sulla differenza lessicale, ma deve intuire dei rapporti più complessi tra le parole del prompt e del target.

La prima rappresentazione utilizzata per la classificazione è quella basata sulle feature linguistiche. Risulta evidente che questo tipo di task sia difficile da risolvere basandosi unicamente sullo stile linguistico. Nonostante ciò, sono stati tentati diversi metodi di combinazione del prompt e del target, tra questi, il migliore è stato il raggruppamento delle feature in diversi gruppi, per poi usare come vettore rappresentativo la cosine similarity tra il prompt e il target per ognuno dei gruppi. Questa rappresentazione, usata per adddestrare una linear SVM, ha riportato un'accuracy del 54%, quindi con un 4% di miglioramento rispetto alla baseline.

La seconda rappresentazione sperimentata, è stata quella basata sugli n-grammi. Sono stati formati quattro dataset diversi: n-grammi di forme, lemmi, POS, e caratteri di inizio o fine parola. Nessuno dei quattro dataset ha portato a dei risultati minimamente positivi; questa rappresentazione quindi, in questo progetto, si è rivelata fallimentare.

La terza rappresentazione, invece, è stata basata su dei vettori di word embedding pre-addestrati attraverso l'algoritmo Word2Vec. Anche in questo caso, è stato fondamentale un approccio di feature engineering per combinare i vettori delle parole in un unico vettore rappresentativo. La rappresentazione migliore è stata ottenuta considerando

per ogni istanza la coppia di vettori di embedding più simili tra quelli del prompt e quelli del target, usando poi come vettore rappresentativo la differenza assoluta tra i due. Con questo approccio, è stato ottenuto un 63% di accuracy sul test set, ottimo risultato, che riesce anche ad avvicinarsi alle prestazioni del fine tuning di BERT, con una maggiore semplicità e velocità di esecuzione.

Infine, per risolvere il task, sono stati eseguiti tre diversi esperimenti di fine tuning sul modello BERT base italiano, distinti per la testa di classificazione utilizzata. Il miglior risultato è stato il 68% di accuracy sul test set del modello con la testa di classificazione personalizzata; ma anche gli altri due esperimenti hanno riportato risultati tutto sommato simili. Per mantenere semplice l'addestramento e per poter più facilmente sviluppare e addestrare la testa di classificazione personalizzata, è stato usato il modello base di BERT, è probabile che, con un modello più grande, i risultati possano essere migliori.

In conclusione, i risultati degli esperimenti sembrano soddisfacenti, ed evidenziano che, per questo tipo di task, se non si utilizza un LLM, è fondamentale una manipolazione ingegnosa delle feature (e del rapporto tra prompt e target) per ottenere dei risultati positivi. Invece, attraverso il finetuning di un LLM, si possono ottenere risultati migliori senza preoccuparsi particolarmente della manipolazione dei dati, ovviamente al prezzo di un costo molto maggiore in termini di risorse computazionali e di tempo.