

RELAZIONE:

Text-Summarizer

Enrico Ferraiolo 0001191698

Laurea Magistrale in Informatica

Corso: Natural Language Processing
a.a. 2024-2025

Indice

1 Introduzione	3
1.1 Struttura del progetto	3
2 Pipeline	4
3 Dataset	4
4 Preprocessing dei Dati	5
4.1 Pulizia del Testo	5
4.2 Filtraggio dei Dati	6
4.3 Tokenizzazione e Token Speciali	7
5 Architettura dei Modelli	7
5.1 Classe Base Astratta	7
5.2 Training	8
5.3 Iperparametri	8
5.3.1 Callbacks	9
5.4 Architetture Sperimentate	9
5.5 Seq2SeqLSTM	10
5.5.1 Training	10
5.5.2 Risultati	11
5.5.3 Architettura	11
5.6 Seq2SeqBiLSTM	11
5.6.1 Training	11
5.6.2 Risultati	12
5.6.3 Architettura	12
5.7 Seq2Seq3BiLSTM	13
5.7.1 Training	13
5.7.2 Risultati	14
5.7.3 Architettura	14
5.8 Seq2SeqGRU	14
5.8.1 Training	14
5.8.2 Risultati	15
5.8.3 Architettura	15
6 Metriche di Valutazione	16
6.1 Valutazione per Permutazione	16
6.2 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)	16
6.3 WER (Word Error Rate)	17
6.4 Cosine Similarity	18
6.5 BERTScore	18
6.6 My Evaluation	19
6.7 Confronto tra le Architetture	20
7 Conclusioni	20
7.1 Considerazioni Finali	20
7.2 Test del Modello Migliore	21

1 Introduzione

L’obiettivo principale del progetto è generare riassunti concisi e significativi a partire da recensioni di prodotti più lunghe, mantenendo il significato del testo originale. Il progetto si articola nelle seguenti fasi:

- **Raccolta e preparazione dei dati:** selezione e pre-elaborazione di un dataset di recensioni di prodotti, con particolare attenzione alla pulizia e alla normalizzazione del testo.
- **Progettazione e implementazione di architetture di reti neurali:** studio e sviluppo di modelli basati su meccanismi di attenzione per la sintesi testuale.
- **Addestramento e inferenza:** realizzazione di pipeline per l’addestramento dei modelli e per l’esecuzione delle operazioni di sintesi su nuovi testi.
- **Valutazione sperimentale:** analisi comparativa delle prestazioni dei modelli mediante metriche standardizzate, al fine di identificare le soluzioni ottimali.

Questo documento vuole illustrare le scelte progettuali e le metodologie adottate per la realizzazione del progetto, nonché i risultati sperimentali ottenuti. *Text-Summarizer* è un framework Python per la sintesi automatica di testi, progettato per essere modulare e facilmente estendibile.

1.1 Struttura del progetto

Il progetto è strutturato in diverse sezioni, ognuna delle quali affronta un aspetto specifico del lavoro svolto.

La cartella principale contiene i file e le cartelle necessarie per l’esecuzione del progetto:

- **architectures:** cartella contenente le implementazioni delle architetture dei modelli di sintesi automatica;
- **report:** cartella contenente il report finale del progetto;
- **results:** cartella contenente i risultati ottenuti durante l’addestramento e la valutazione dei modelli;
- **requirements.txt:** file contenente le librerie necessarie per l’esecuzione del progetto;
- **text_summarizer_training.ipynb:** notebook Python contenente il codice per l’addestramento e la valutazione dei modelli;
- **text_summarizer_inference.ipynb:** notebook Python contenente il codice per l’inferenza dei modelli.

2 Pipeline

Di seguito, in figura 1, è presentata la pipeline del progetto, che illustra i vari passaggi e le fasi di elaborazione dei dati.

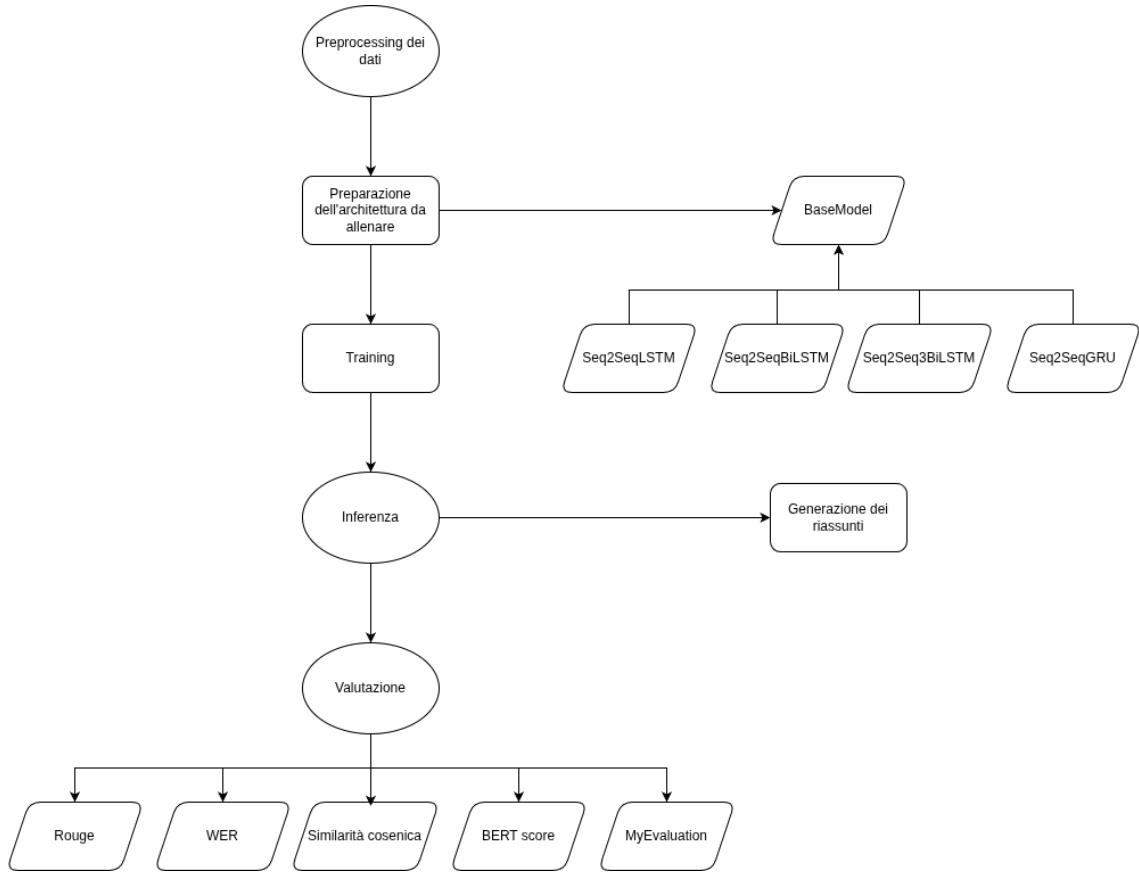


Figura 1: Pipeline del progetto

La pipeline è suddivisa in tre fasi principali:

1. **Preprocessing:** in questa fase vengono eseguite le operazioni di pulizia e preparazione dei dati, come la rimozione di stopwords, la tokenizzazione e la normalizzazione del testo.
2. **Training:** in questa fase viene addestrato il modello di summarization utilizzando i dati preprocessati. Viene effettuata una suddivisione tra training set e validation set.
3. **Evaluation:** in questa fase vengono valutati i riassunti generati dal modello utilizzando diverse metriche, come ROUGE, BERTScore e la metrica personalizzata myevaluation.

Nelle sezioni successive, verranno approfonditi i dettagli di ciascuna fase della pipeline.

3 Dataset

Per questo progetto è stato utilizzato il dataset *SNAP Amazon Fine Food Reviews*[1][3], che contiene recensioni di prodotti alimentari di Amazon in lingua inglese.

In particolare, il dataset contiene, per ogni riga, una recensione completa e il rispettivo riassunto.

Del dataset originale, composto da circa 500.000 righe, è stato selezionato un sottinsieme di 10.000 righe per l'analisi e l'allenamento del modello.

4 Preprocessing dei Dati

Il preprocessing dei dati è una fase critica per garantire la qualità e l'efficacia dei modelli di summarization, infatti è fondamentale pulire e filtrare i dati in modo accurato.

Sul dataset, per l'appunto, sono stati eseguiti diversi passaggi di pulizia e filtraggio dei dati per garantire qualità e coerenza del modello durante l'addestramento.

Vediamo di seguito gli step effettuati durante questa fase:

4.1 Pulizia del Testo

Sono stati applicati i seguenti step di preprocessing:

1. Conversione del testo in minuscolo

- Questa conversione garantisce l'uniformità del testo, evitando che la stessa parola venga considerata diversa solo per la presenza di maiuscole. Ad esempio, "Home", "HOME" e "home" vengono trattate come la stessa parola, riducendo la dimensionalità del vocabolario e migliorando l'efficienza dell'addestramento.

2. Rimozione dei tag HTML

- Le recensioni potrebbero contenere tag HTML residui dal formato web originale. Questi elementi non contribuiscono al significato semantico del testo e potrebbero interferire con l'apprendimento del modello, pertanto vengono rimossi.

3. Espansione delle contrazioni

- Le contrazioni nella lingua inglese (come "don't", "I'm", "we're") vengono espanso nelle loro forme complete ("do not", "I am", "we are"). Questo processo vuole standardizzare e garantire coerenza in tutto il testo e aiuta il modello a catturare meglio le relazioni semantiche, eliminando variazioni non necessarie della stessa espressione.

4. Rimozione degli apostrofi possessivi ('s)

- La forma possessiva in inglese non altera sostanzialmente il significato della frase ai fini del riassunto. La sua rimozione semplifica il testo e riduce ulteriormente la dimensione del vocabolario, permettendo al modello di concentrarsi sui concetti principali.

5. Eliminazione del testo tra parentesi

- Il testo tra parentesi spesso contiene informazioni supplementari che non sono generalmente essenziali per il riassunto.
La loro rimozione aiuta a mantenere il focus sulle informazioni principali della recensione.

6. Rimozione della punteggiatura e caratteri speciali

- La punteggiatura e i caratteri speciali, pur essendo importanti per la leggibilità umana, possono introdurre rumore nell'addestramento del modello.
La loro rimozione semplifica il testo mantenendo intatto il contenuto semantico essenziale per la generazione del riassunto.

7. Eliminazione delle stopwords

- Le stopwords sono parole molto comuni (come "the", "is", "at", "which") che appaiono frequentemente ma portano poco significato semantico.
La loro rimozione riduce significativamente la dimensionalità del problema senza perdere informazioni cruciali per il riassunto, permettendo al modello di concentrarsi sulle parole più significative.

8. Rimozione delle parole troppo corte

- Le parole molto corte (solitamente di una o due lettere) spesso non contribuiscono al significato del testo.
La loro rimozione aiuta a ridurre ulteriormente il rumore nei dati, mantenendo solo i termini più significativi per l'analisi.

4.2 Filtraggio dei Dati

Dopo l'analisi statistica del dataset, sono stati applicati i seguenti vincoli:

- Lunghezza massima delle recensioni: 30 parole
- Lunghezza massima dei riassunti: 8 parole

Questi limiti sono stati determinati attraverso un'analisi statistica della distribuzione delle lunghezze nel dataset, come possiamo vedere nella figura 2.

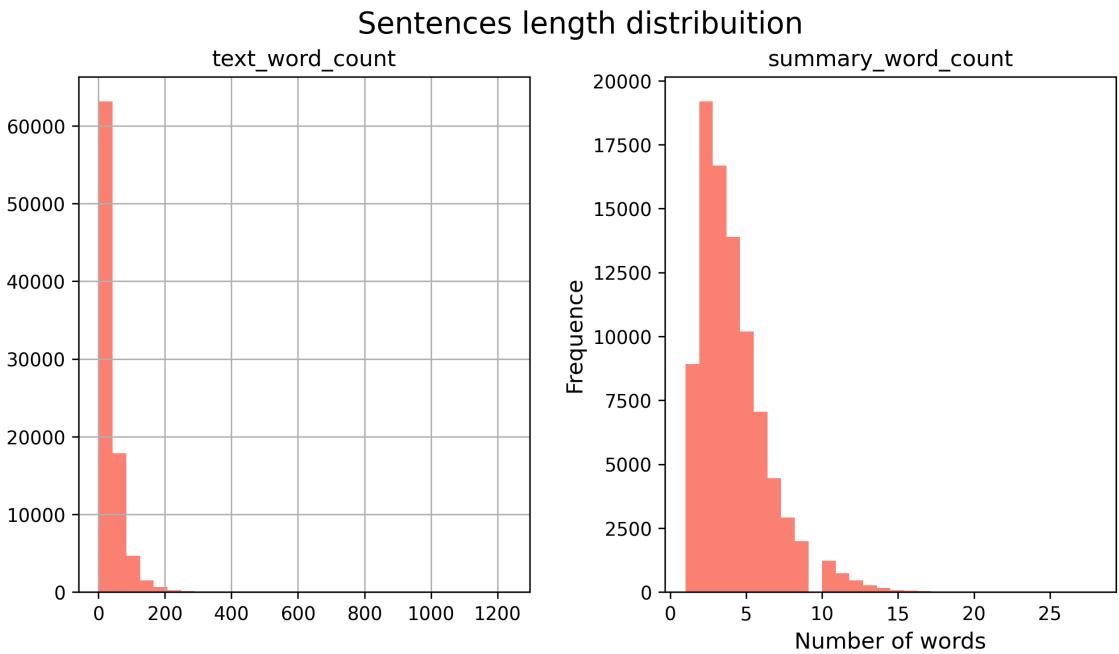


Figura 2: A sinistra la distribuzione delle lunghezze delle recensioni, a destra la distribuzione delle lunghezze dei riassunti

Infatti, come si può notare dai due grafici, la maggior parte delle recensioni e dei riassunti ha lunghezze inferiori ai limiti stabiliti, quindi questi vincoli permettono di mantenere la maggior parte dei dati del dataset.

4.3 Tokenizzazione e Token Speciali

Per preparare i dati per i modelli ho aggiunto i token speciali "`sostok`" e "`eostok`" per indicare l'inizio e la fine di una sequenza, in modo da facilitare la tokenizzazione e la fase di addestramento.

Inoltre, ho effettuato la tokenizzazione separatamente per le recensioni (testo di input) e i riassunti (testo di output) per garantire che il modello possa apprendere correttamente la relazione tra i due. I due tokenizer servono a creare il vocabolario per le recensioni e per i riassunti, in modo da poter convertire i testi in sequenze di token.

5 Architettura dei Modelli

Per svolgere questo progetto ho deciso di effettuare un confronto tra diversi modelli, implementando architetture Seq2Seq con LSTM e GRU.

L'implementazione dei modelli è stata effettuata attraverso una classe astratta `BaseModel` e la successiva creazioni e implementazione di classi derivate.

Questo permette di definire un'interfaccia comune per tutti i modelli di summarization e di estendere facilmente l'architettura in futuro.

5.1 Classe Base Astratta

La classe `BaseModel` fornisce l'interfaccia base per tutti i modelli di summarization:

- Metodi astratti per costruire encoder e decoder.
- Funzionalità per il salvataggio, caricamento e inferenza del modello.
- Conversione tra sequenze e testo tramite i tokenizer.

5.2 Training

L’addestramento dei modelli, derivati dalla classe `BaseModel`, è stato effettuato utilizzando il dataset preprocessato.

Prima di iniziare l’addestramento, il dataset è stato suddiviso in training set e validation set, con una proporzione del 90% e 10% rispettivamente.

Dopodiché sono passato alla fase effettiva di training dei modelli, utilizzando e la loss function `Sparse Categorical Crossentropy`, utile nei task di summarization.

5.3 Iperparametri

Per ogni classe di modello, il training è stato eseguito più volte, ciascuna con una combinazione diversa di iperparametri.

Queste combinazioni vengono generate tramite una *hyperparameter_grid* implementata nella funzione `create_hyperparameter_grid`, che restituisce tutte le permutazioni possibili tra i valori forniti per:

- `embedding_dim`: dimensione dell’embedding delle parole
- `hidden_dim`: dimensione dello stato nascosto dell’encoder e del decoder
- `encoder_dropout`: tasso di dropout dell’encoder
- `encoder_recurrent_dropout`: tasso di dropout per gli stati ricorrenti dell’encoder
- `decoder_dropout`: tasso di dropout del decoder
- `decoder_recurrent_dropout`: tasso di dropout per gli stati ricorrenti del decoder
- `optimizer`: ottimizzatore da utilizzare durante l’addestramento
- `learning_rate`: tasso di apprendimento
- `batch_size`: dimensione del batch durante l’addestramento
- `epochs`: numero di epoche per l’addestramento

Per ogni **permutazione** di iperparametri, viene eseguito il seguente processo di allenamento:

1. **Preparazione dei dati:** viene eseguita la parte di caricamento e preprocessing dei dati
2. **Istanza del modello:** viene istanziata la classe del modello corrente con gli iperparametri selezionati

3. **Compilazione del modello:** viene compilato il modello ed eseguita la fase di training con diverse callback, di cui parleremo nella sezione successiva
4. **Training del modello:** viene eseguito il training del modello
5. **Salvataggio dei risultati:**

- *Pesi del modello* (`result/{model_class}/weights/`)
- *Architettura del modello* (`result/{model_class}/media/architectures/`)
- *Grafico dell'andamento della loss* (`result/{model_class}/media/graphs/`)
- *Cronologia dell'allenamento* (`result/{model_class}/histories/`)
- *Riassunti generati:* al termine del training vengono generati i riassunti (dal validation set) e salvati in un file csv (`result/{model_class}/csv/`)

5.3.1 Callbacks

Durante il training ho utilizzato anche le seguenti funzioni di callback:

- **Early Stopping:** monitora una metrica, in questo caso la validation loss, e interrompe l'addestramento se non ci sono miglioramenti per un certo numero di epoche consecutive. Questo aiuta a prevenire l'overfitting e a risparmiare tempo di calcolo.
- **Learning Rate Scheduler:** regola il tasso di apprendimento durante il training secondo una strategia, nel mio caso ho utilizzato la **Step Decay**, che riduce il learning rate di un fattore fisso ogni tot epoche.
- **Reduce LR on Plateau:** monitora una metrica, in questo caso la validation loss, e riduce il learning rate se non ci sono miglioramenti per un certo numero di epoche consecutive. Questo aiuta a ottimizzare il processo di addestramento e a trovare un tasso di apprendimento più efficace.

5.4 Architetture Sperimentate

Sono state sperimentate diverse architetture di modelli di summarization, ognuna con caratteristiche e parametri diversi.

Le due categorie principali di modelli implementati sono:

- **LSTM:** modelli basati su layer LSTM per encoder e decoder.
- **GRU:** modelli basati su layer GRU per encoder e decoder.

Tali architetture sono basate sulle RNN (Recurrent Neural Networks) e sono state scelte per la loro efficacia nei task di text-summarization, poiché gestiscono le dipendenze tra le parole su sequenze di testo.

- **LSTM:** Long Short-Term Memory, è una variante delle RNN che risolve il problema del vanishing gradient, grazie alla presenza di un meccanismo di memoria a lungo termine. Tale meccanismo di gating permette di memorizzare informazioni importanti e scartare quelle meno rilevanti.

- **GRU**: Gated Recurrent Unit, è una variante più semplice delle LSTM, con meno parametri e meno complessità computazionale. Anche in questo caso, il meccanismo di gating permette di memorizzare informazioni importanti e scartare quelle meno rilevanti.

Al fine di rendere più scorrevole la lettura, per ogni classe vengono riportati solamente i migliori risultati ottenuti durante l’addestramento con i migliori parametri e le migliori configurazioni trovate (basate sul **BERTScore**, di cui verrà parlato in seguito), sebbene siano stati effettuati numerosi tentativi e test riportati in seguito in una tabella comparativa.

5.5 Seq2SeqLSTM

La classe **Seq2SeqLSTM** implementa l’architettura specifica per il modello di summarization Sequence to Sequence con layer LSTM.

5.5.1 Training

L’addestramento del modello è stato effettuato attraverso la seguente configurazione:

Parametro	Valore
Ottimizzatore	Adam
Learning rate	0.001
Embedding dimension	512
Latent dimension	128
Decoder dropout	0.2
Decoder recurrent dropout	0.2
Encoder dropout	0.2
Encoder recurrent dropout	0.2
Batch size	128
Epochs	50

Tabella 1: Configurazione dell’addestramento del modello

Possiamo verificare l’andamento delle loss durante l’addestramento nella figura 3.

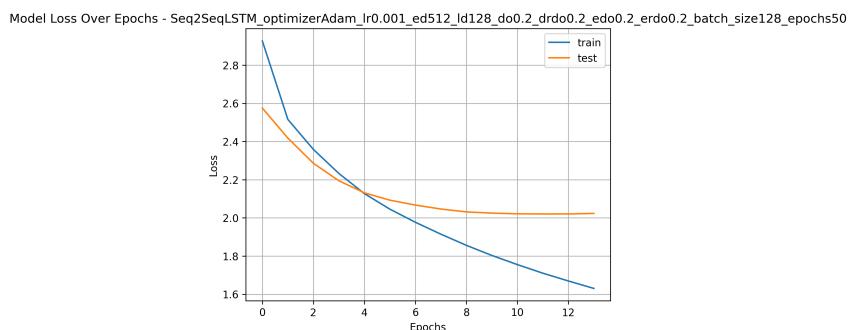


Figura 3: Andamento della *loss* e *validation loss* durante l’addestramento

5.5.2 Risultati

Questo modello ha ottenuto i seguenti risultati al termine dell’addestramento:

- **Loss:** 1.63
- **Validation loss:** 2.02
- **Accuracy:** 0.67
- **Validation Accuracy:** 0.64

5.5.3 Architettura

L’architettura utilizzata è la seguente:

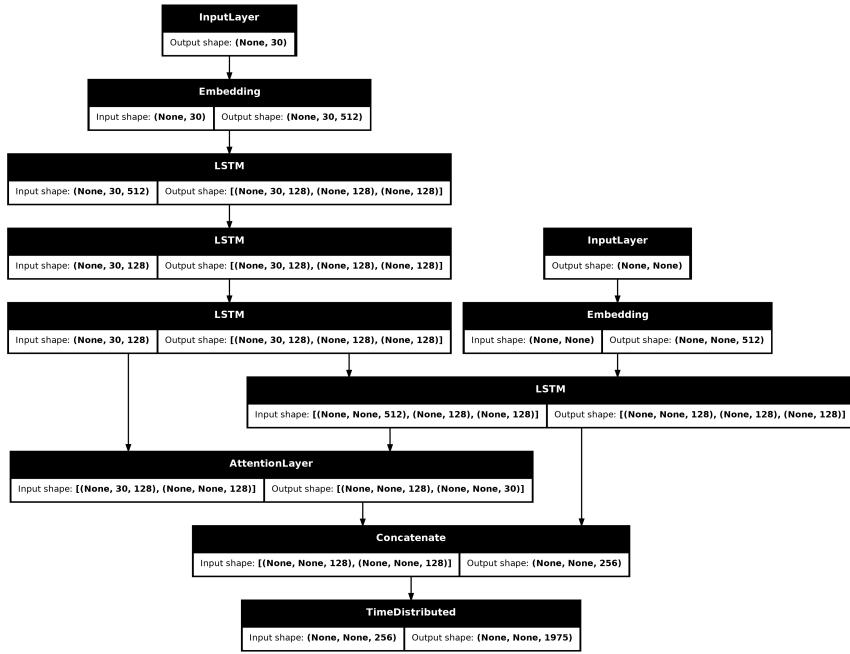


Figura 4: Architettura del modello Seq2SeqLSTM

5.6 Seq2SeqBiLSTM

La classe Seq2SeqBiLSTM implementa un’architettura simile al modello Seq2SeqLSTM, ma i layer LSTM dell’encoder sono bidirezionali.

5.6.1 Training

L’addestramento del modello è stato effettuato attraverso la seguente configurazione:

Possiamo verificare l’andamento delle loss durante l’addestramento nella figura 5.

Parametro	Valore
Ottimizzatore	Adam
Learning rate	0.001
Embedding dimension	512
Latent dimension	256
Decoder dropout	0.2
Decoder recurrent dropout	0.2
Encoder dropout	0.2
Encoder recurrent dropout	0.2
Batch size	128
Epochs	50

Tabella 2: Configurazione dell’addestramento del modello

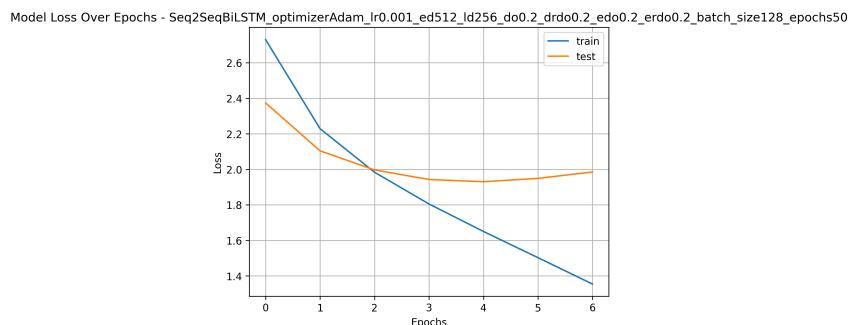


Figura 5: Andamento della *loss* e *validation loss* durante l’addestramento

5.6.2 Risultati

Questo modello ha ottenuto i seguenti risultati al termine dell’addestramento:

- **Loss:** 1.35
- **Validation loss:** 1.98
- **Accuracy:** 0.70
- **Validation Accuracy:** 0.65

5.6.3 Architettura

L’architettura utilizzata è la seguente:

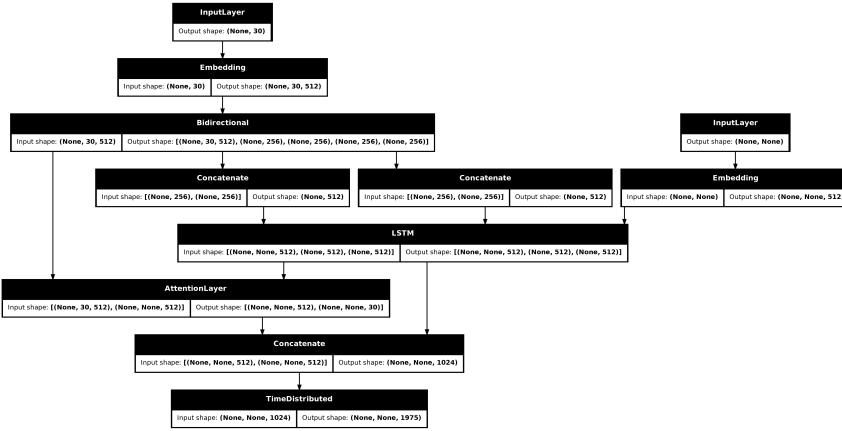


Figura 6: Architettura del modello Seq2SeqBiLSTM

5.7 Seq2Seq3BiLSTM

La classe Seq2Seq3BiLSTM implementa un'architettura simile al modello Seq2SeqBiLSTM, ma con tre layer LSTM bidirezionali nell'encoder.

5.7.1 Training

L'addestramento del modello è stato effettuato attraverso la seguente configurazione:

Parametro	Valore
Ottimizzatore	Adam
Learning rate	0.001
Embedding dimension	256
Latent dimension	256
Decoder dropout	0.2
Decoder recurrent dropout	0.2
Encoder dropout	0.2
Encoder recurrent dropout	0.2
Batch size	128
Epochs	50

Tabella 3: Configurazione dell'addestramento del modello

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 7.

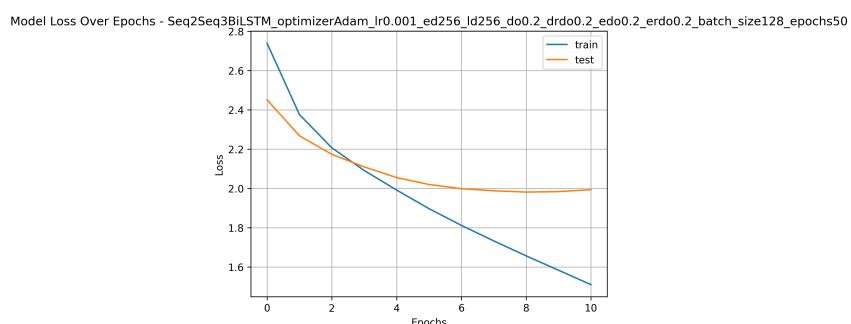


Figura 7: Andamento della *loss* e *validation loss* durante l'addestramento

5.7.2 Risultati

Questo modello ha ottenuto i seguenti risultati al termine dell'addestramento:

- **Loss:** 1.51
- **Validation loss:** 1.99
- **Accuracy:** 0.68
- **Validation Accuracy:** 0.65

5.7.3 Architettura

L'architettura utilizzata è la seguente:

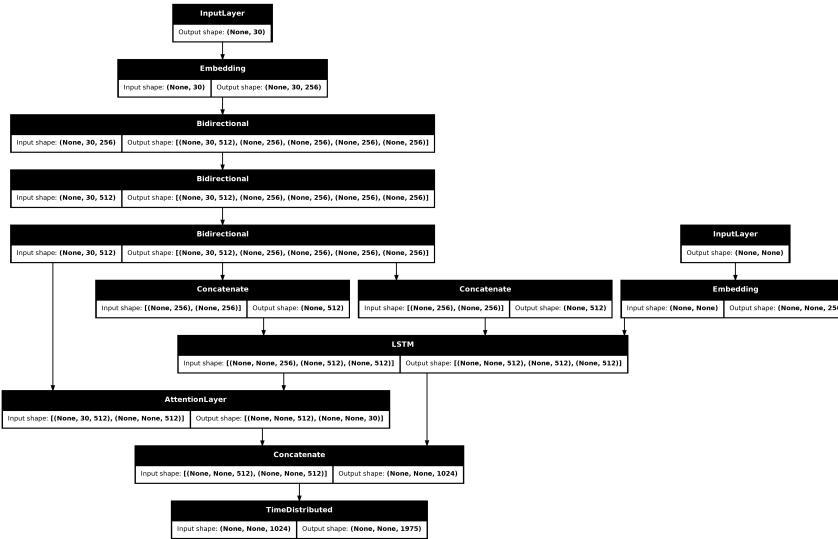


Figura 8: Architettura del modello Seq2Seq3BiLSTM

5.8 Seq2SeqGRU

La classe Seq2SeqGRU implementa un'architettura Seq2Seq con GRU, utilizzando layer GRU per l'encoder e il decoder.

5.8.1 Training

L'addestramento del modello è stato effettuato attraverso la seguente configurazione:

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 9.

Parametro	Valore
Ottimizzatore	Adam
Learning rate	0.001
Embedding dimension	256
Latent dimension	128
Decoder dropout	0.2
Decoder recurrent dropout	0.2
Encoder dropout	0.2
Encoder recurrent dropout	0.2
Batch size	128
Epochs	50

Tabella 4: Configurazione dell’addestramento del modello

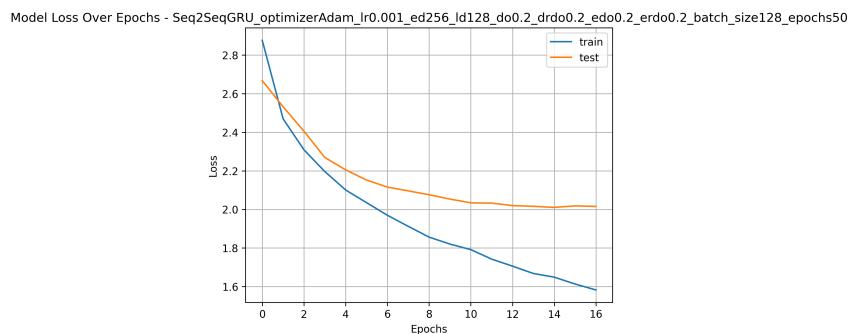


Figura 9: Andamento della *loss* e *validation loss* durante l’addestramento

5.8.2 Risultati

Questo modello ha ottenuto i seguenti risultati al termine dell’addestramento:

- **Loss:** 1.58
- **Validation loss:** 2.01
- **Accuracy:** 0.67
- **Validation Accuracy:** 0.64

5.8.3 Architettura

L’architettura utilizzata è la seguente:

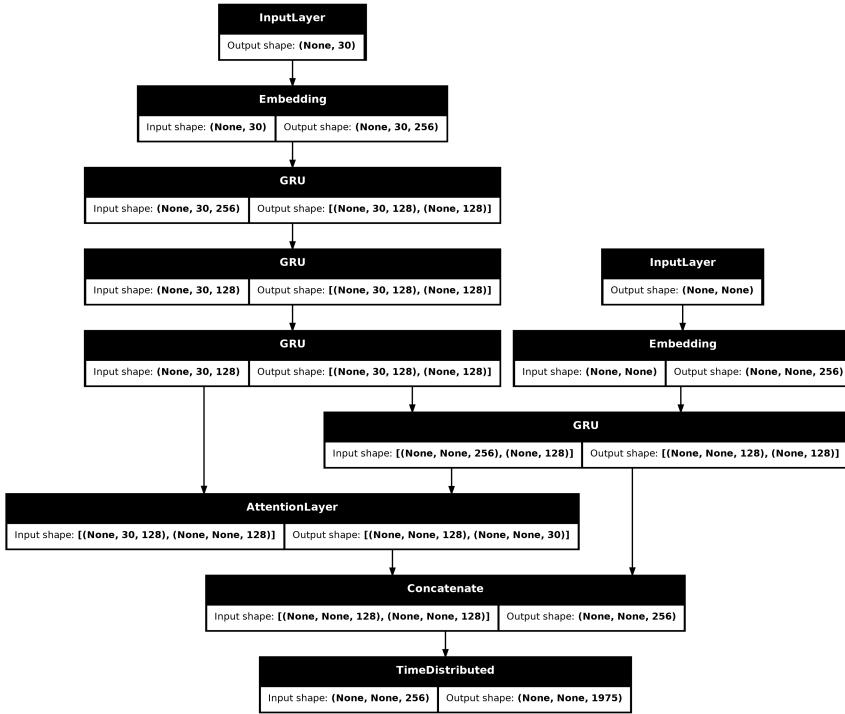


Figura 10: Architettura del modello Seq2SeqGRU

6 Metriche di Valutazione

In questa sezione vengono analizzate le prestazioni dei modelli attraverso i grafici delle principali metriche di valutazione: ROUGE [2], WER [4], cosine similarity e BERTScore [5].

Inoltre è stata anche utilizzata la metrica personalizzata `myevaluation`, che verrà approfondita in seguito.

Queste metriche sono state scelte per misurare in modo accurato la qualità dei riassunti generati, confrontandoli con quelli di riferimento.

Le valutazioni sono state effettuate utilizzando il dataset di test, che non è stato utilizzato durante l'addestramento del modello.

6.1 Valutazione per Permutazione

Per ogni permutazione delle architetture e dei relativi iperparametri, è stato seguito un processo di valutazione strutturato al termine della fase di addestramento. Di seguito sono riportate le metriche di valutazione utilizzate.

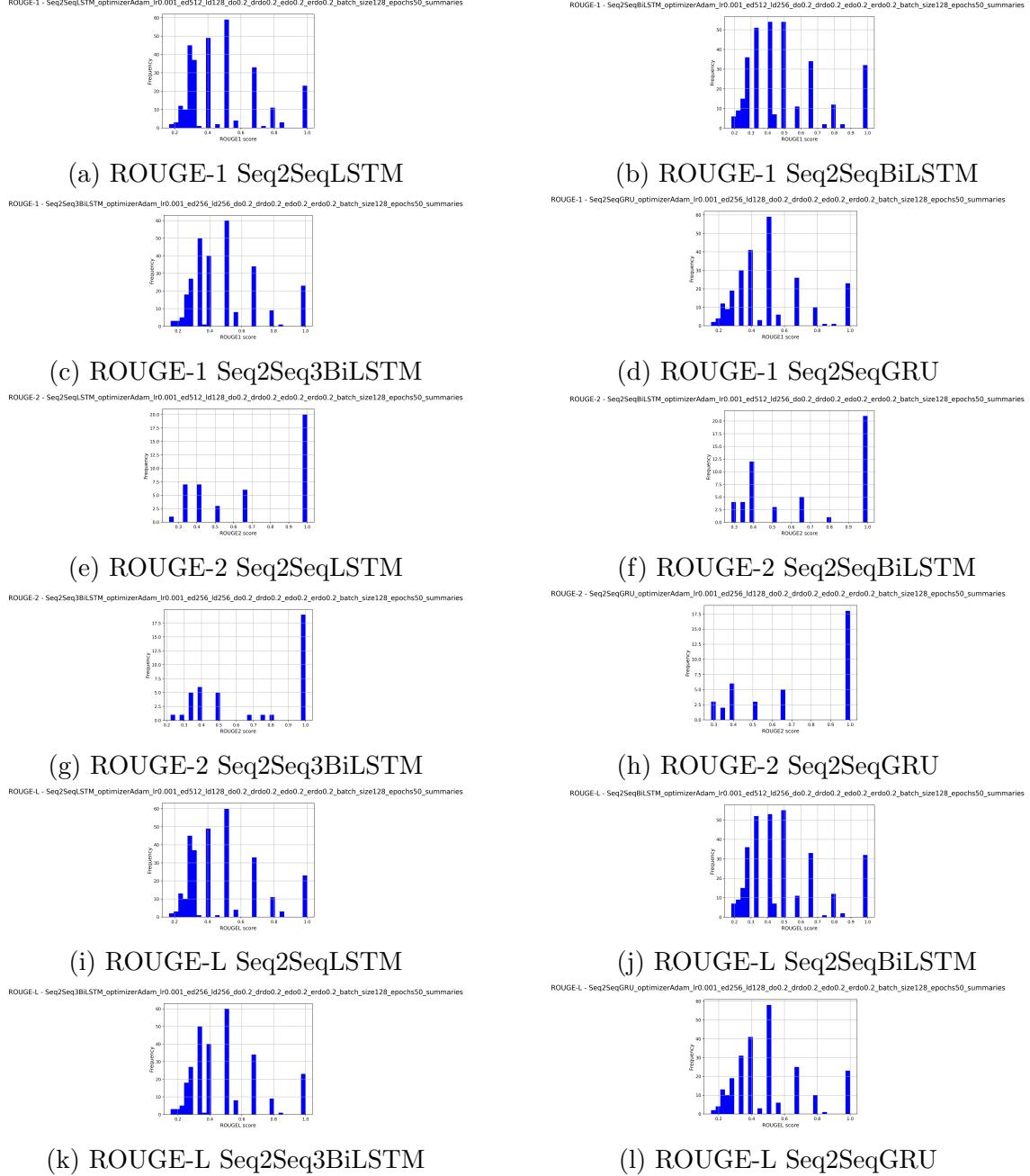
6.2 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Sono state calcolate tre varianti di ROUGE:

- ROUGE-1: confronta unigrammi tra il riassunto generato e quello di riferimento
- ROUGE-2: considera bigrammi per valutare la similarità tra i due testi

- ROUGE-L: confronta la sottosequenza più lunga comune tra i due testi

I grafici nella Figura 11 confrontano le performance in termini di ROUGE-1, ROUGE-2 e ROUGE-L per i quattro modelli. Il modello Seq2SeqBiLSTM mostra un miglioramento nei punteggi ROUGE rispetto al Seq2SeqLSTM, Seq2Seq3BiLSTM e Seq2SeqGRU, indicando una maggiore capacità di catturare similarità lessicali.



6.3 WER (Word Error Rate)

Il WER è una metrica che calcola il tasso di errore tra due sequenze di parole. In particolare, il WER calcola il numero di operazioni di inserimento, cancellazione e sostituzione necessarie per trasformare una sequenza di parole in un'altra.

Il confronto del WER, mostrato nella Figura 12, evidenzia che il modello Seq2Seq3BiLSTM ottiene risultati migliori, indicando una maggiore accuratezza nella generazione delle parole, nonostante la differenza con l'architettura Seq2SeqBiLSTM non sia significativa.

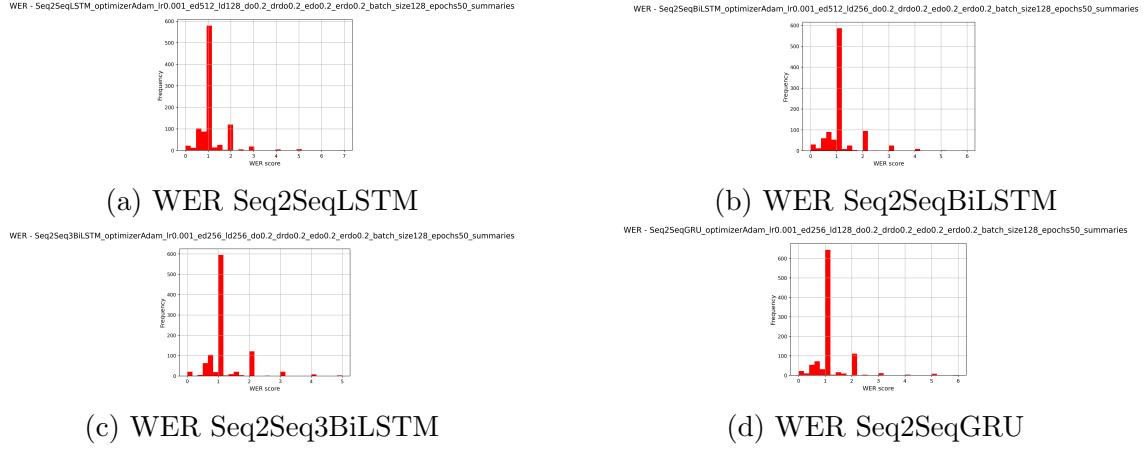


Figura 12: Confronto del Word Error Rate tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM e Seq2SeqGRU.

6.4 Cosine Similarity

La similarità cosenica è una metrica che calcola la similarità tra due vettori in uno spazio multidimensionale.

Nel caso specifico della generazione di riassunti, la similarità cosenica è stata calcolata tra i vettori di embedding delle parole nei riassunti generati e quelli nei riassunti di riferimento, con il fine di valutare la qualità dei riassunti generati.

La Figura 13 confronta i valori di similarità cosenica. Anche in questo caso, il modello Seq2SeqBiLSTM ottiene valori più alti, suggerendo una maggiore correlazione semantica con i riassunti di riferimento.

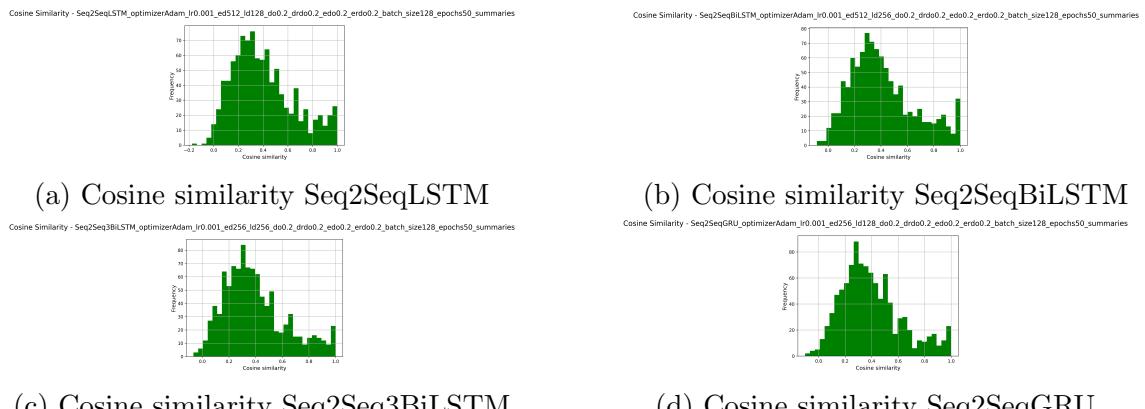


Figura 13: Confronto della cosine similarity tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM e Seq2SeqGRU.

6.5 BERTScore

Il BERTScore è una metrica che utilizza il modello BERT per calcolare la similarità tra due testi.

In particolare, il BERTScore calcola la similarità tra i vettori di embedding delle parole nei riassunti generati e quelli nei riassunti di riferimento, tenendo conto del contesto semantico delle parole.

La Figura 14 mostra i valori di BERTScore per i diversi modelli. Anche in questo caso, il modello Seq2SeqBiLSTM ottiene risultati migliori, suggerendo una maggiore capacità di catturare la similarità semantica tra i riassunti generati e quelli di riferimento.

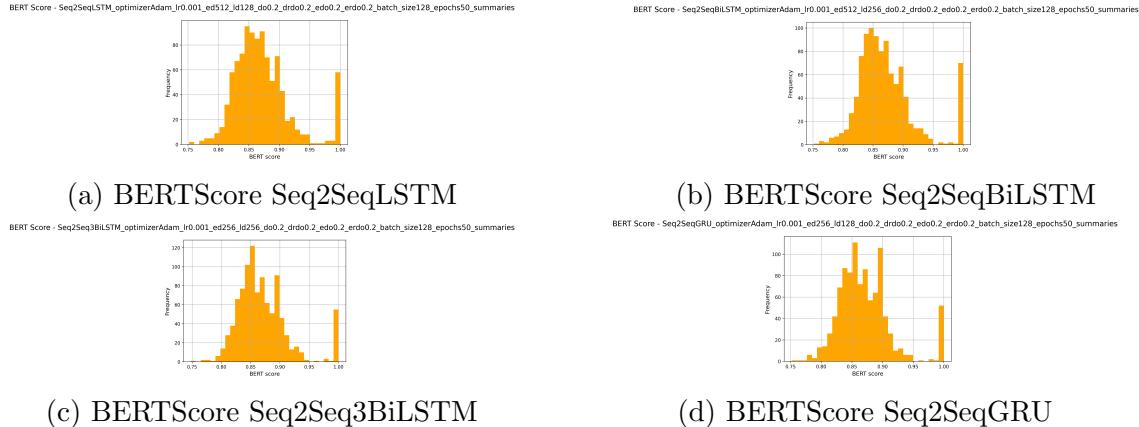


Figura 14: Confronto del BERTScore tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM e Seq2SeqGRU.

6.6 My Evaluation

La metrica `myevaluation` è stata progettata per valutare la qualità dei riassunti generati in modo più dettagliato.

Essa tiene conto di diversi fattori e ciascuno di essi ha uno specifico peso in base alla sua importanza.

La formula con cui viene calcolata la metrica `myevaluation` è la seguente:

$$\text{MyEval} = cs_PS_OS \cdot W_{cs_PS_OS} + keyword_overlap \cdot W_{keyword_overlap} + bert_score \cdot W_{bert_score}$$

Dove:

- cs_PS_OS è la similarità cosenica tra il riassunto generato e quello originale
- $keyword_overlap$ è la percentuale di parole chiave presenti nel riassunto generato rispetto a quelle nel riassunto originale
- $bert_score$ è il punteggio BERTScore tra il riassunto generato e quello originale
- $W_{cs_PS_OS}$, $W_{keyword_overlap}$ e W_{bert_score} sono i pesi associati a ciascun fattore
 - $W_{cs_PS_OS} = 0.07$
 - $W_{keyword_overlap} = 0.03$
 - $W_{bert_score} = 0.9$

La Figura 15 mostra i risultati della metrica `myevaluation` per i diversi modelli. Il modello Seq2SeqBiLSTM ottiene i punteggi più alti, suggerendo una qualità superiore dei riassunti generati rispetto agli altri modelli.

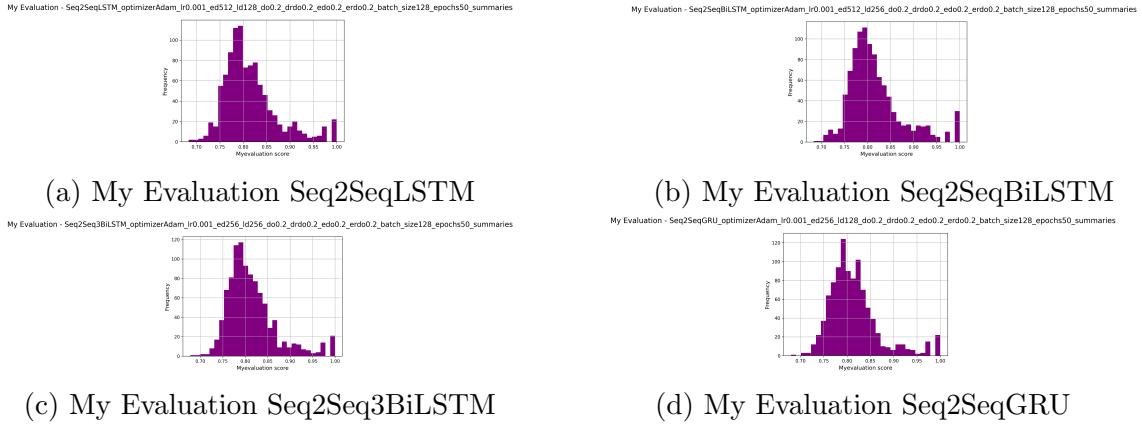


Figura 15: Confronto della metrica `myevaluation` tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM e Seq2SeqGRU.

6.7 Confronto tra le Architetture

Di seguito viene riportata una tabella comparativa (figura 16) tra i vari modelli implementati, con i rispettivi risultati ottenuti al termine dell’addestramento.

Per ogni classe di modello sono stati effettuati numerosi tentativi e test, riportati in seguito in una tabella comparativa le varie istanze di configurazione e i risultati ottenuti.

Possiamo subito notare che il modello Seq2SeqBiLSTM ha ottenuto i risultati migliori in quasi tutte le metriche.

Model - Instance	mean_cosine	mean_myevaluation	mean_BERTScore	mean_wer	mean_rouge1	mean_rouge2	mean_rougeL
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed256_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3917	0.8117	0.0680	1.1509	0.1405	0.0264	0.1394
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed256_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3935	0.8136	0.0702	1.1316	0.1358	0.0283	0.1358
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed512_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3944	0.8114	0.0677	1.1882	0.1360	0.0306	0.1357
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed512_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3961	0.8130	0.0694	1.1676	0.1326	0.0313	0.1321
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed256_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4073	0.8147	0.0695	1.1524	0.1538	0.0334	0.1516
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed256_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4017	0.8146	0.0701	1.1152	0.1538	0.0350	0.1534
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed512_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4014	0.8135	0.0685	1.1508	0.1548	0.0335	0.1536
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed512_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4107	0.8165	0.0711	1.0993	0.1596	0.0339	0.1589
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed256_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3972	0.8142	0.0707	1.1241	0.1222	0.0268	0.1210
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed256_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4049	0.8143	0.0696	1.1448	0.1474	0.0310	0.1468
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed512_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4108	0.8153	0.0703	1.1671	0.1416	0.0310	0.1406
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed512_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4054	0.8147	0.0698	1.1207	0.1541	0.0313	0.1531
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed256_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3906	0.8134	0.0703	1.1592	0.1272	0.0293	0.1262
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed256_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.3979	0.8132	0.0693	1.1364	0.1344	0.0281	0.1340
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed512_ld128_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4013	0.8152	0.0710	1.1357	0.1410	0.0309	0.1406
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed512_ld256_doo2_drd02_edo02_erd02_batch_size128_epochs50_summaries	0.4018	0.8122	0.0678	1.1454	0.1405	0.0287	0.1397

Figura 16: Comparazione delle istanze dei modelli

7 Conclusioni

7.1 Considerazioni Finali

Il lavoro svolto ha portato allo sviluppo di un sistema di sintesi automatica in grado di generare riassunti coerenti a partire da recensioni di cibo. Il sistema è stato progettato per garantire facilità di estensione e adattabilità a diverse tipologie di task, aumentando così la sua versatilità e potenziale applicativo in contesti variegati.

Nel corso del progetto sono stati implementati diversi modelli di sintesi automatica, tra cui Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM e Seq2SeqGRU. I risultati sono stati valutati attraverso numerose metriche, quali ROUGE, Word Error Rate (WER), Cosine Similarity, BERTScore e MyEvaluation. In particolare, la BERTScore si è rivelata la metrica più rappresentativa della qualità dei riassunti,

in quanto considera il contesto semantico e la similarità delle parole.

Inoltre, la metrica **MyEvaluation** ha fornito una valutazione più dettagliata della qualità dei riassunti, integrando vari fattori e assegnando a ciascuno un peso in base alla sua rilevanza. I risultati sperimentali indicano che il modello **Seq2SeqBiLSTM** ha ottenuto le prestazioni migliori su quasi tutte le metriche, suggerendo una superiore capacità di catturare le similarità semantiche tra i riassunti generati e quelli di riferimento.

È opportuno sottolineare che i risultati sono stati influenzati dalla scelta del dataset e dalle specifiche caratteristiche dei dati, inclusi gli aspetti relativi al preprocessing. Tali elementi evidenziano l'importanza di un'attenta selezione e preparazione dei dati per ottimizzare le prestazioni dei modelli di sintesi automatica.

7.2 Test del Modello Migliore

Il modello **Seq2SeqBiLSTM** ha mostrato prestazioni superiori rispetto agli altri modelli implementati.

Di seguito vengono riportati alcuni esempi di riassunti generati dal modello **Seq2SeqBiLSTM** e i rispettivi riassunti di riferimento.

----- REVIEW -----	
Review:	one dog fav glad get amazon
Original summary:	my puppy loves these
Predicted summary:	my dog loves this
----- REVIEW -----	
Review:	took trip bar runs good gluten eating friends family packed snacks every morning yum
Original summary:	great energy bar
Predicted summary:	great for gluten free
----- REVIEW -----	
Review:	absolutely delicious ranks right tuna excitement category year old cat wish organic
Original summary:	so says my spoiled cat
Predicted summary:	delicious
----- REVIEW -----	
Review:	product one month use date bottles lot try use month still one best products used
Original summary:	very good product
Predicted summary:	great product

Figura 17: Esempio di riassunto generato dal modello Seq2SeqBiLSTM

Riferimenti bibliografici

- [1] *Amazon Fine Food Reviews*. <https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews>. Visitato il: 21/05/2025. 2025.
- [2] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, lug. 2004, pp. 74–81. URL: <https://aclanthology.org/W04-1013>.
- [3] Julian McAuley e Jure Leskovec. *From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews*. 2013. arXiv: [1303.4402 \[cs.SI\]](https://arxiv.org/abs/1303.4402). URL: <https://arxiv.org/abs/1303.4402>.
- [4] Somnath Roy. *Semantic-WER: A Unified Metric for the Evaluation of ASR Transcript for End Usability*. 2021. arXiv: [2106.02016 \[cs.CL\]](https://arxiv.org/abs/2106.02016). URL: <https://arxiv.org/abs/2106.02016>.
- [5] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: [1904.09675 \[cs.CL\]](https://arxiv.org/abs/1904.09675). URL: <https://arxiv.org/abs/1904.09675>.