

# **RELAZIONE:**

## **Text-Summarizer**

Enrico Ferraiolo 0001191698

**Laurea Magistrale in Informatica**

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

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Dataset</b>	<b>3</b>
<b>3</b>	<b>Preprocessing dei Dati</b>	<b>3</b>
3.1	Pulizia del Testo . . . . .	3
3.2	Filtraggio dei Dati . . . . .	5
3.3	Tokenizzazione e Token Speciali . . . . .	5
<b>4</b>	<b>Architettura dei Modelli</b>	<b>5</b>
4.1	Classe Base Astratta . . . . .	6
4.2	Training . . . . .	6
4.2.1	Callbacks . . . . .	6
4.3	Architetture sperimentate . . . . .	6
4.4	LSTM . . . . .	6
4.5	Seq2SeqLSTM . . . . .	6
4.5.1	Encoder . . . . .	7
4.5.2	Decoder . . . . .	7
4.5.3	Training . . . . .	8
4.5.4	Risultati . . . . .	8
4.6	Seq2SeqBiLSTM . . . . .	9
4.6.1	Encoder . . . . .	9
4.6.2	Decoder . . . . .	9
4.6.3	Training . . . . .	10
4.6.4	Risultati . . . . .	10
4.7	Seq2Seq3BiLSTM . . . . .	11
4.7.1	Training . . . . .	12
4.7.2	Risultati . . . . .	12
4.8	Seq2SeqLSTMGloVe . . . . .	12
4.8.1	Training . . . . .	12
4.8.2	Risultati . . . . .	13
4.9	Seq2SeqLSTMTrasformer . . . . .	13
4.9.1	Encoder . . . . .	13
4.9.2	Decoder . . . . .	14
4.9.3	Training . . . . .	15
4.9.4	Risultati . . . . .	16
<b>5</b>	<b>GRU</b>	<b>16</b>
5.1	Confronto tra le Architetture . . . . .	17
<b>6</b>	<b>Metriche di Valutazione</b>	<b>17</b>
6.1	ROUGE (Recall-Oriented Understudy for Gisting Evaluation) . . . . .	17
6.2	WER (Word Error Rate) . . . . .	19
6.3	Cosine Similarity . . . . .	19
<b>7</b>	<b>Conclusioni</b>	<b>20</b>

# 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.

## 2 Dataset

Per questo progetto è stato utilizzato il dataset [SNAP Amazon Fine Food Reviews](#), 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.

## 3 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:

### 3.1 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.

### 3.2 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 1.

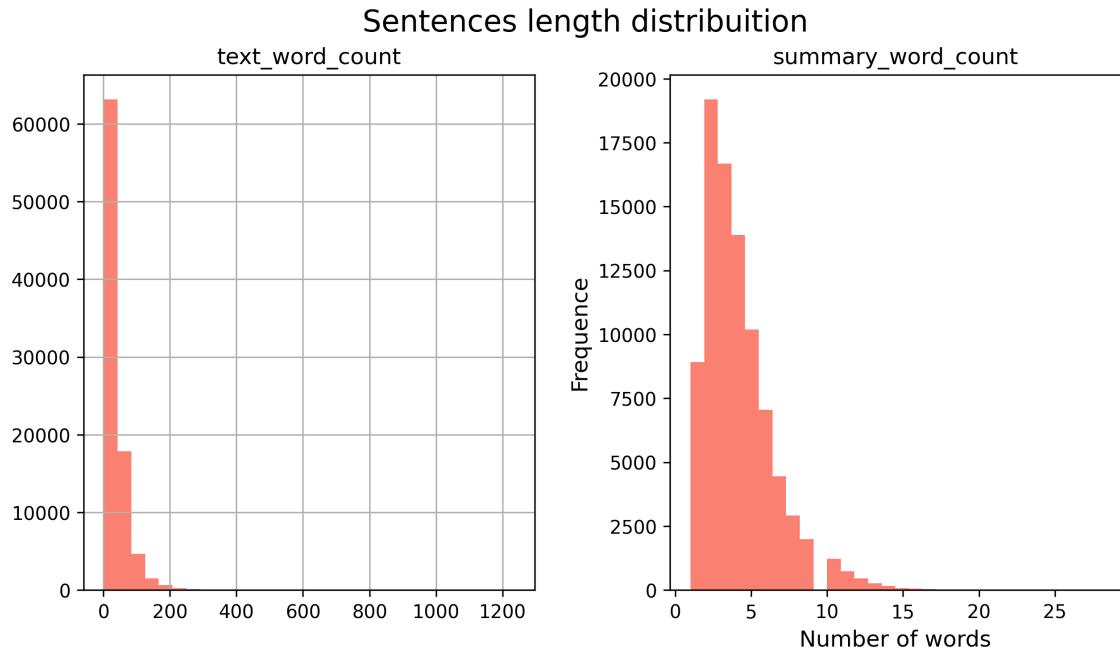


Figura 1: A sinistra la distribuzione delle lunghezze delle recensioni, a destra la distribuzione delle lunghezze dei riassunti

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.

### 3.3 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.

## 4 Architettura dei Modelli

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.

## 4.1 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.

## 4.2 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 la loss function `Sparse Categorical Crossentropy`, utile nei task di summarization.

### 4.2.1 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.

## 4.3 Architetture sperimentate

### 4.4 LSTM

### 4.5 Seq2SeqLSTM

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

Vediamo di seguito le caratteristiche principali dell’architettura, composta da encoder e decoder:

#### 4.5.1 Encoder

L'encoder è composto da:

- **Layer di embedding:** mappa i token di input in vettori di lunghezza fissa
- **Tre layer LSTM** con:
  - Dimensione latente fissa
  - Dropout del 40%
  - Recurrent dropout del 20%

#### 4.5.2 Decoder

Il decoder include:

- **Layer di embedding:** mappa i token di output in vettori di lunghezza fissa
- **Layer LSTM** con:
  - Stessa dimensione latente dell'encoder
  - Dropout del 40%
  - Recurrent dropout del 20%
- **Layer di attention:** calcola i pesi di attenzione tra l'encoder e il decoder
- **Layer denso di output:** questo layer restituisce la distribuzione di probabilità sul vocabolario per la generazione delle parole del riassunto.  
Utilizza la funzione di attivazione softmax per la normalizzazione delle probabilità.

Di seguito, nella figura 2, possiamo vedere un diagramma dell'architettura del modello:

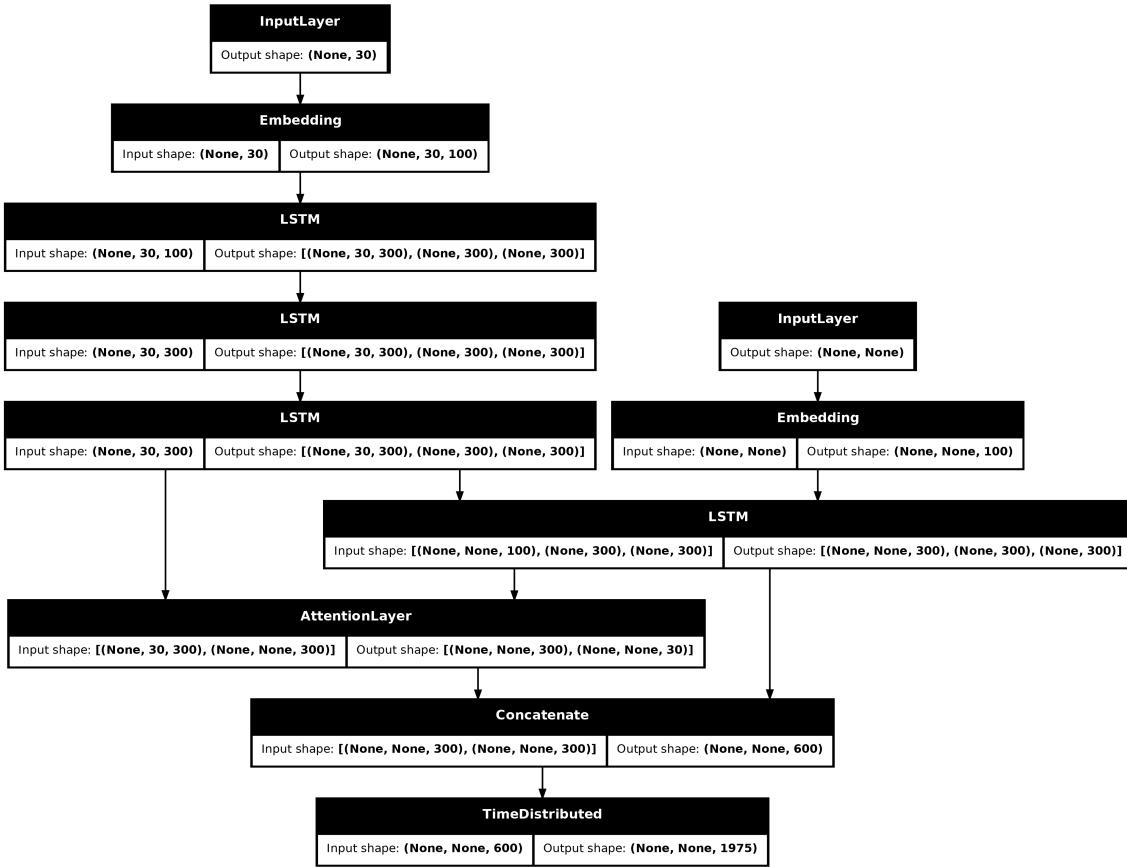


Figura 2: Diagramma dell’architettura del modello Seq2SeqLSTM

#### 4.5.3 Training

L’addestramento del modello è stato effettuato con ottimizzatore Adam e 50 epocha con early stopping.

#### 4.5.4 Risultati

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

- **Loss:** DA INSERIRE
- **Validation loss:** DA INSERIRE
- **Accuracy:** DA INSERIRE
- **Validation Accuracy:** DA INSERIRE

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

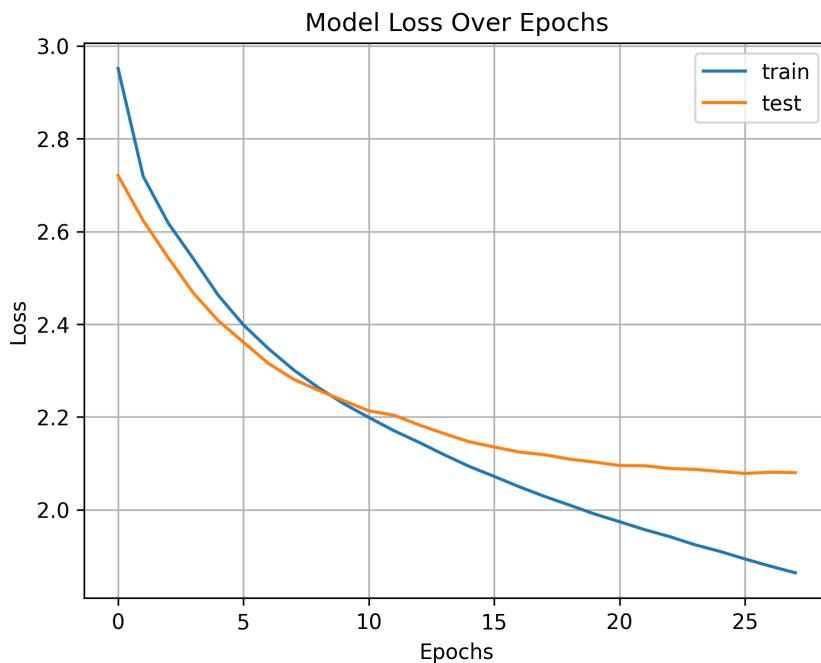


Figura 3: Andamento delle loss durante l’addestramento

## 4.6 Seq2SeqBiLSTM

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

### 4.6.1 Encoder

L’encoder è composto da:

- **Layer di embedding**
- **Un layer LSTM bidirezionale** con:
  - Dimensione latente fissa
  - Dropout del 40%
  - Recurrent dropout del 40%
- **Layer di Concatenazione:** Concatena gli stati del layer LSTM bidirezionale

### 4.6.2 Decoder

Il decoder include:

- **Layer di embedding**
- **Layer LSTM:**
  - Dimensione latente pari  $2 * dimensione\_latente\_fissa$ , per adattarsi alla concatenazione degli stati provenienti dall’encoder.
  - Dropout del 40%
  - Recurrent dropout del 20%

- Layer di attenzione
- Layer di Concatenazione
- Layer denso di output

Di seguito, possiamo vedere un diagramma dell'architettura del modello Seq2SeqBiLSTM nella figura 4.

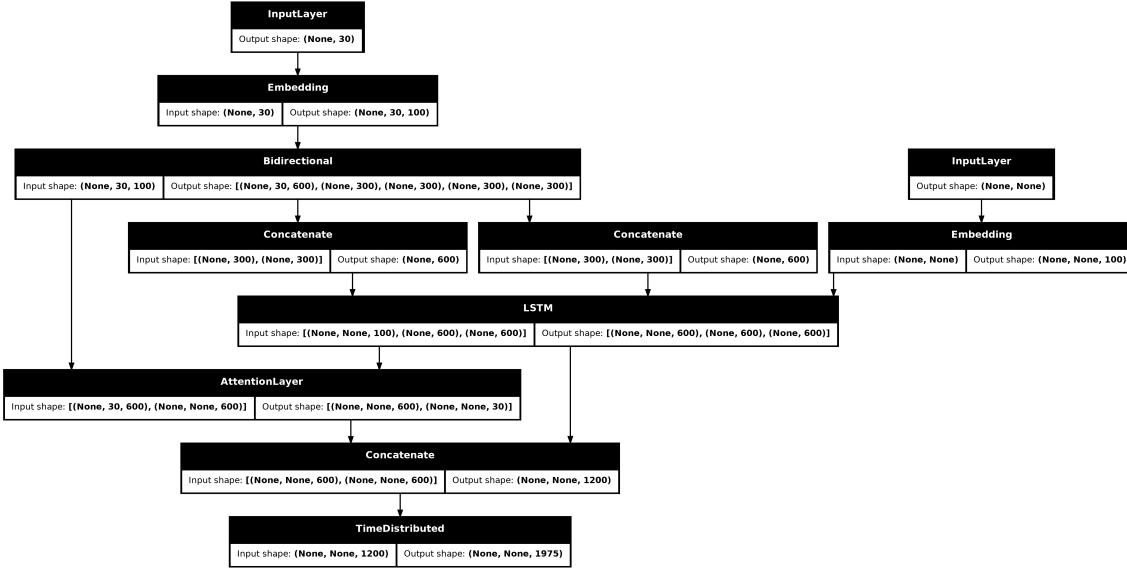


Figura 4: Diagramma dell'architettura del modello Seq2SeqBiLSTM

#### 4.6.3 Training

L'addestramento del modello è stato effettuato con ottimizzatore Adam e 50 epocha con early stopping.

#### 4.6.4 Risultati

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

- **Loss:** DA INSERIRE
- **Validation loss:** DA INSERIRE
- **Accuracy:** DA INSERIRE
- **Validation Accuracy:** DA INSERIRE

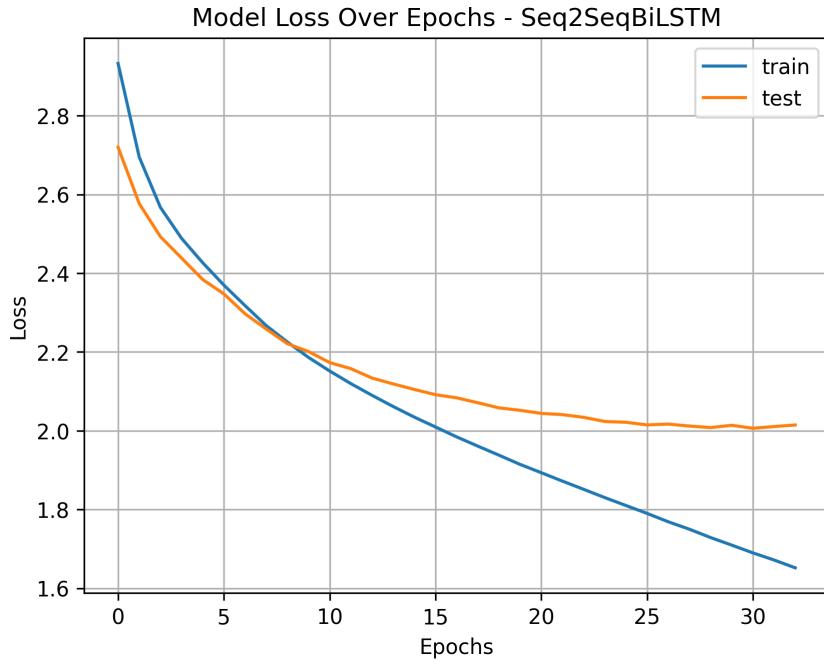


Figura 5: Andamento delle loss durante l’addestramento

## 4.7 Seq2Seq3BiLSTM

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

Di seguito, possiamo vedere un diagramma dell’architettura del modello Seq2Seq3BiLSTM nella figura 6.



Figura 6: Diagramma dell’architettura del modello Seq2Seq3BiLSTM

#### 4.7.1 Training

L’addestramento del modello è stato effettuato con ottimizzatore Adam e 50 epoche con early stopping.

#### 4.7.2 Risultati

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

- **Loss:** DA INSERIRE
- **Validation loss:** DA INSERIRE
- **Accuracy:** DA INSERIRE
- **Validation Accuracy:** DA INSERIRE

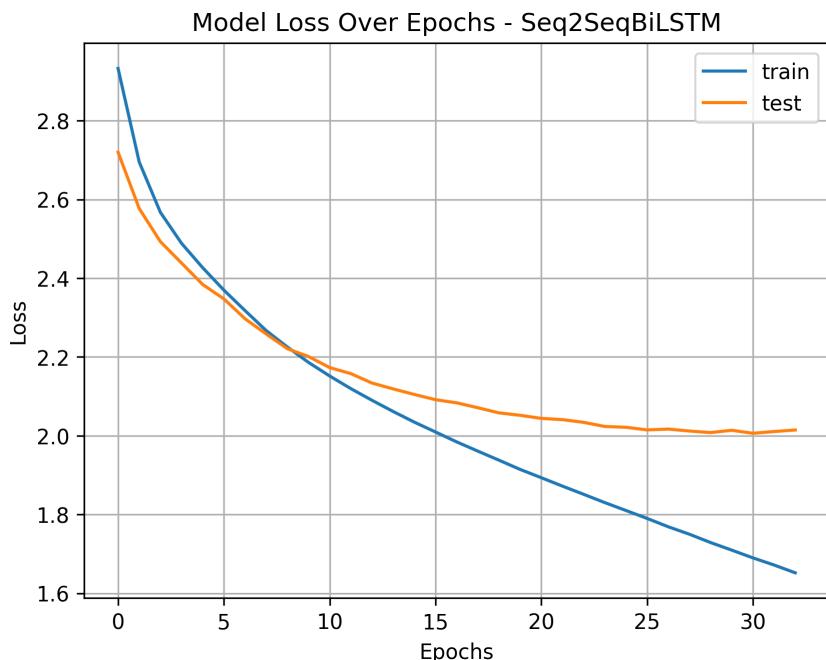


Figura 7: Andamento delle loss durante l’addestramento

### 4.8 Seq2SeqLSTMGloVe

La classe `Seq2SeqLSTMGloVe` implementa un’architettura simile al modello Seq2SeqLSTM, utilizzando i vettori di embedding GloVe preaddestrati per la rappresentazione delle parole.

Più precisamente vengono scaricati e utilizzati i vettori di embedding GloVe preaddestrati da [Stanford NLP Group](#) da 100 dimensioni, anche se la classe consente di scambiare facilmente i vettori con quelli di dimensione diversa.

#### 4.8.1 Training

L’addestramento del modello è stato effettuato con ottimizzatore Adam e 50 epoche con early stopping.

#### 4.8.2 Risultati

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

- **Loss:** DA INSERIRE
- **Validation loss:** DA INSERIRE
- **Accuracy:** DA INSERIRE
- **Validation Accuracy:** DA INSERIRE

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 8.

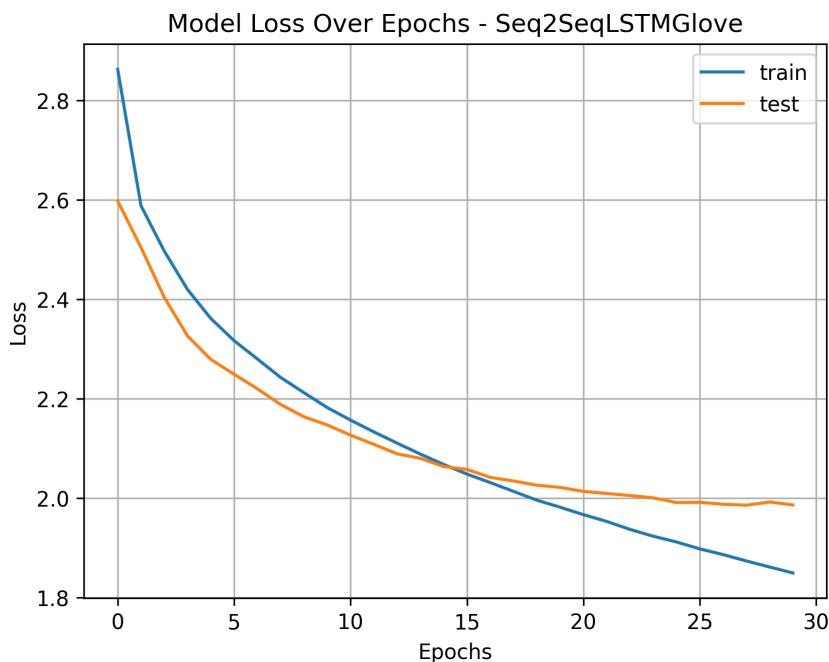


Figura 8: Andamento delle loss durante l'addestramento

#### 4.9 Seq2SeqLSTMTrasformer

La classe `Seq2SeqLSTMTrasformer` implementa un'architettura simile al modello Seq2SeqLSTM, ma inoltre aggiunge 2 blocchi Transformer sia nell'encoder che nel decoder.

Parametri totali: 7,801,623

##### 4.9.1 Encoder

- Layer di embedding
- Due Blocchi Transformer con:
  - Dimensione latente pari a 300
  - Dropout del 40%
- Tre layer LSTM con:

- Dimensione latente pari a 300
- Dropout del 40%
- Recurrent dropout del 40%

#### 4.9.2 Decoder

- **Layer di embedding**
- **Due blocchi Transformer** con:
  - Dimensione latente pari a 300
  - Dropout del 40%
  - Recurrent dropout del 20%
  - Un MultiHeadAttention da 8 Head
  - Un layer di Dropout del 10%
  - Un Add layer
  - Un layer di Normalizzazione
  - Tre layer di FeedForward con:
    - \* Un layer denso con attivazione ReLU e dimensione latente pari a 512
    - \* Un layer denso con dimensione latente pari a 300
    - \* Un layer di Dropout del 10%
    - \* Un Add layer
    - \* Un layer di Normalizzazione
- **Layer LSTM** con:
  - Stessa dimensione latente dell'encoder
  - Dropout del 40%
  - Recurrent dropout del 40%
- **Layer di attention**
- **Layer denso di output**

Di seguito, possiamo vedere un diagramma dell'architettura del modello Seq2SeqLSTMTrasformer nella figura 9.

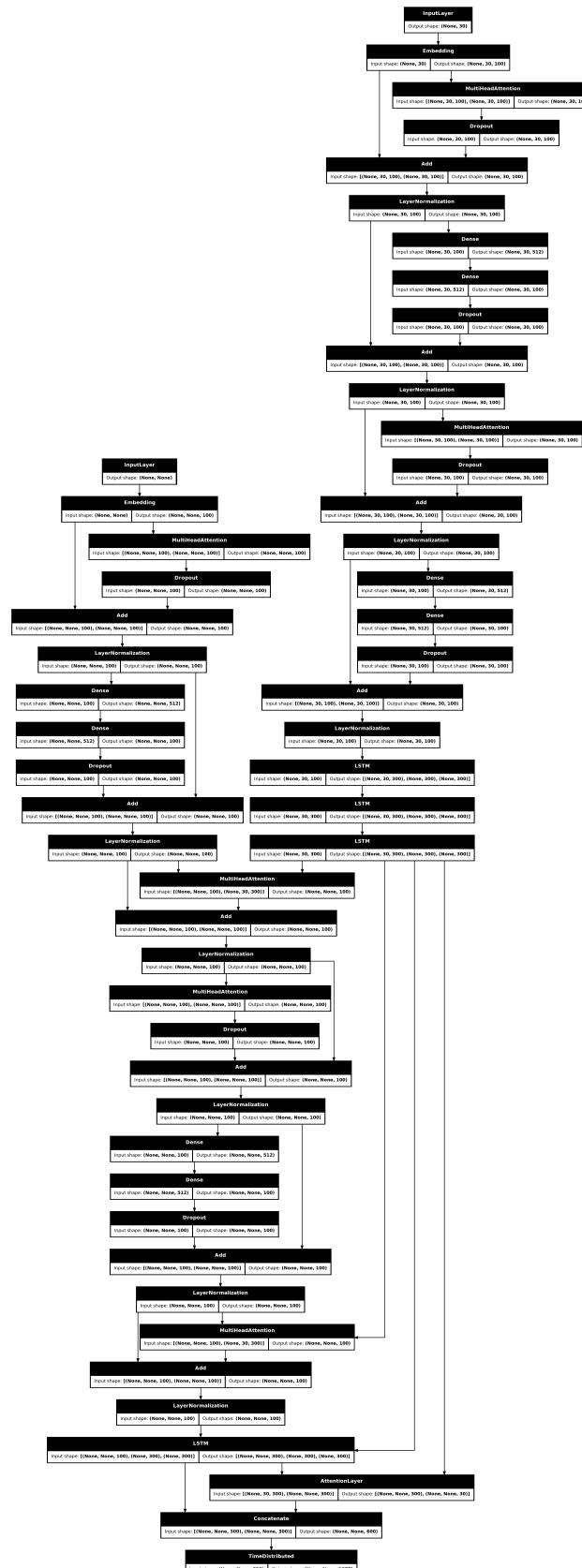


Figura 9: Diagramma dell'architettura del modello Seq2SeqLSTMTransformer

#### 4.9.3 Training

L'addestramento del modello è stato effettuato con ottimizzatore Adam e 50 epocha con early stopping.

#### 4.9.4 Risultati

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

- **Loss:** 0.2015
- **Validation loss:** 0.5034
- **Accuracy:** 0.9576
- **Validation Accuracy:** 0.9269

Possiamo verificare l'andamento della funzione di loss durante l'addestramento del modello nella figura 10.

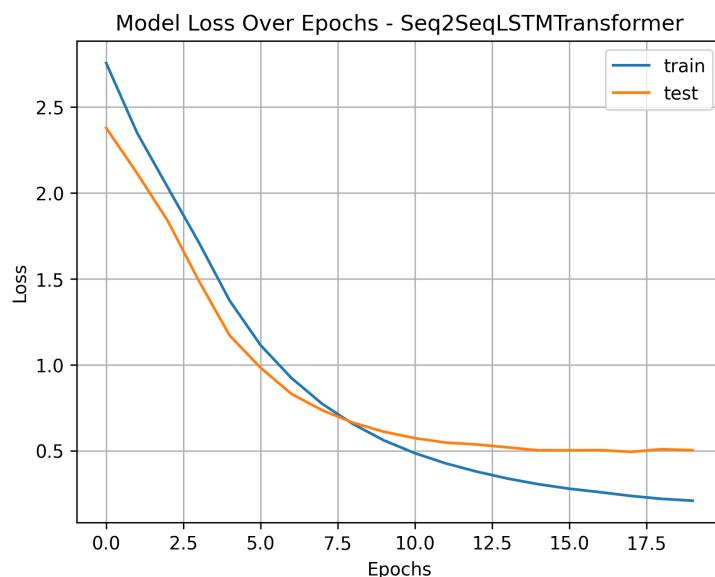


Figura 10: Andamento della funzione di loss durante l'addestramento del modello Seq2SeqLSTMTrasformer

|||||| HEAD

## 5 GRU

=====

## 5.1 Confronto tra le Architetture

Model - Instance	mean_cosine	mean_myevaluation	mean_wer	mean_rouge1	mean_rouge2	mean_rougeL
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size128_epochs50_summaries	0.4189	0.4189	1.1440	0.1564	0.0411	0.1552
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size256_epochs50_summaries	0.4318	0.4318	1.1382	0.1665	0.0435	0.1653
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size64_epochs50_summaries	0.4198	0.4198	1.1611	0.1588	0.0383	0.1583
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr.001_ed512_d256_d0.2_drd0.2_ed0.2_batch_size128_epochs50_summaries	0.4631	0.4631	1.0647	0.2240	0.0696	0.2227
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size256_epochs50_summaries	0.4431	0.4431	0.0895	0.2075	0.0575	0.2052
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr.001_ed512_d256_d0.2_drd0.2_ed0.2_batch_size64_epochs50_summaries	0.4731	0.4731	1.0187	0.2416	0.0738	0.2397
Seq2SeqBiLSTMImproved - Seq2SeqBiLSTMImproved_optimizerAdam_lr.001_ed300_ld256_d0.3_drd0.3_ed0.3_batch_size64_epochs50_summaries	0.1034	0.1034	1.1710	0.0136	0.0000	0.0136
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size128_epochs50_summaries	0.4294	0.4294	1.0916	0.1656	0.0417	0.1650
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size256_epochs50_summaries	0.3899	0.3899	1.1170	0.1290	0.0255	0.1287
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size64_epochs50_summaries	0.4445	0.4445	1.1173	0.1884	0.0563	0.1875
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr.001_ed512_d256_d0.2_drd0.2_ed0.2_batch_size128_epochs50_summaries	0.4400	0.4400	1.0826	0.1702	0.0453	0.1694
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size256_epochs50_summaries	0.4374	0.4374	1.0963	0.1629	0.0407	0.1616
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr.001_ed512_d256_d0.2_drd0.2_ed0.2_batch_size64_epochs50_summaries	0.4266	0.4266	1.1331	0.1607	0.0454	0.1604
Seq2SeqLSTMGlove - Seq2SeqLSTMGlove_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size128_epochs50_summaries	0.4314	0.4314	1.0877	0.1798	0.0472	0.1795
Seq2SeqLSTMGlove - Seq2SeqLSTMGlove_optimizerAdam_lr.001_ed512_ld256_d0.2_drd0.2_ed0.2_batch_size256_epochs50_summaries	0.4467	0.4467	1.0803	0.1927	0.0562	0.1918
Seq2SeqLSTMGlove - Seq2SeqLSTMGlove_optimizerAdam_lr.001_ed512_d256_d0.2_drd0.2_ed0.2_batch_size64_epochs50_summaries	0.4378	0.4378	1.1131	0.1861	0.0519	0.1855

Figura 11: Comparazione delle istanze dei modelli

l*l**l**l**l**l**l* 08b0ca8 (feat: update report with architecture comparison and additional figure)

## 6 Metriche di Valutazione

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

Durante la fase di inferenza, sono stati generati 1000 riassunti a partire dai dati di test.

### 6.1 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 12 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 e al Seq2SeqLSTMGlove, indicando una maggiore capacità di catturare similarità lessicali.

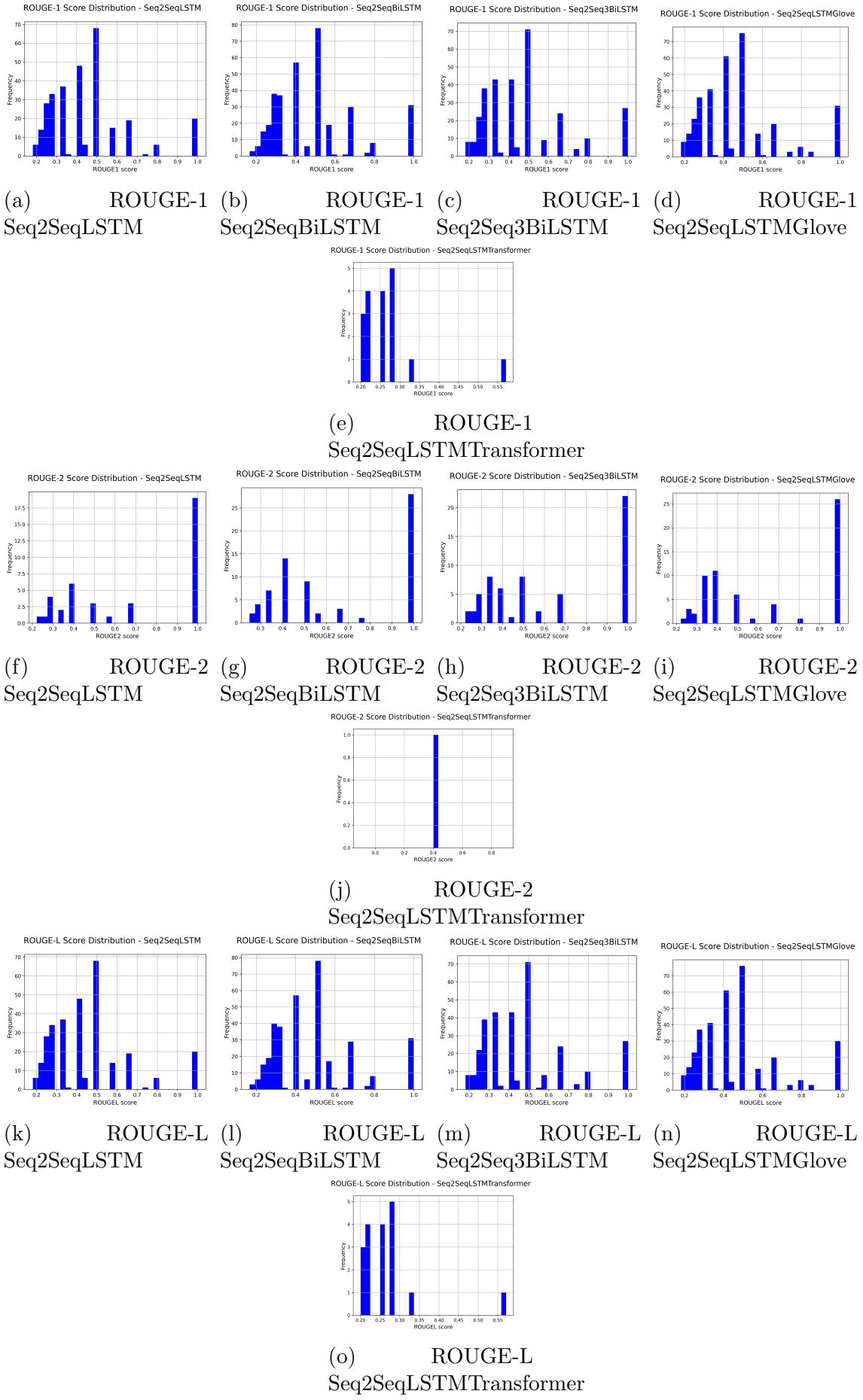


Figura 12: Confronto dei punteggi ROUGE tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM, Seq2SeqLSTMGlove e Seq2SeqLSTMTransformer.

## 6.2 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 13, evidenzia che il modello Seq2SeqBiLSTM ottiene risultati migliori, indicando una maggiore accuratezza nella generazione delle parole.

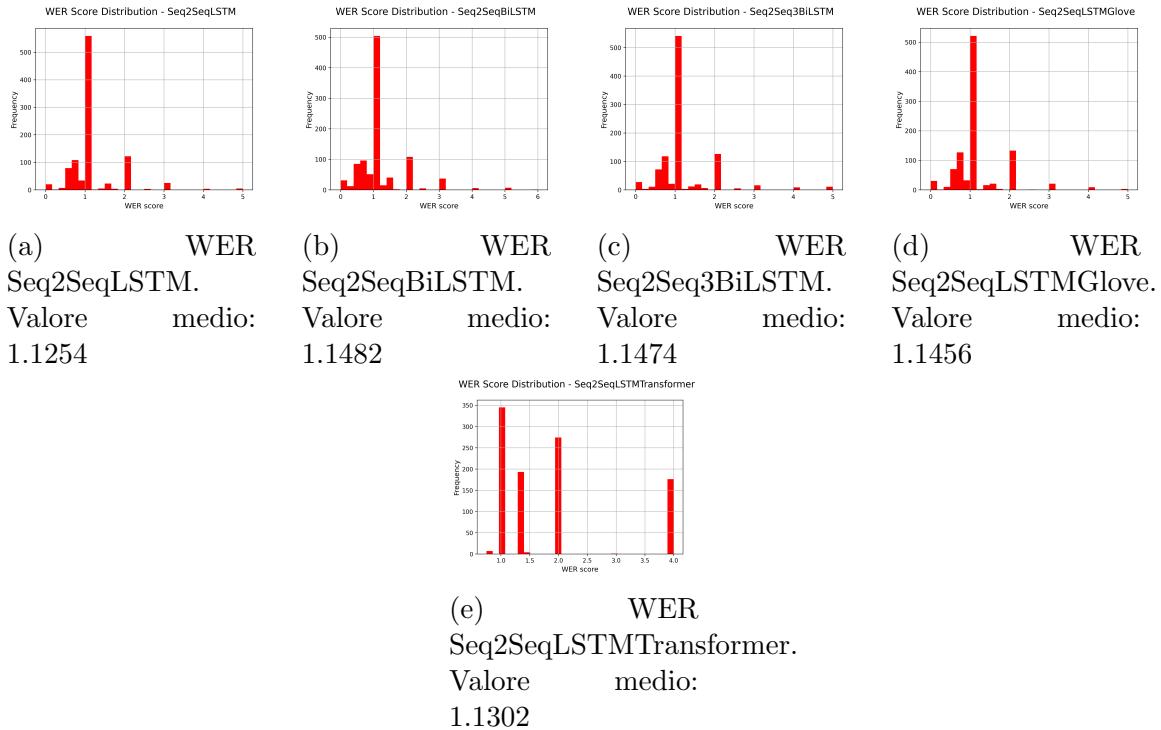


Figura 13: Confronto del Word Error Rate tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM, Seq2SeqLSTMGlove e Seq2SeqLSTMTransformer.

## 6.3 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 14 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.

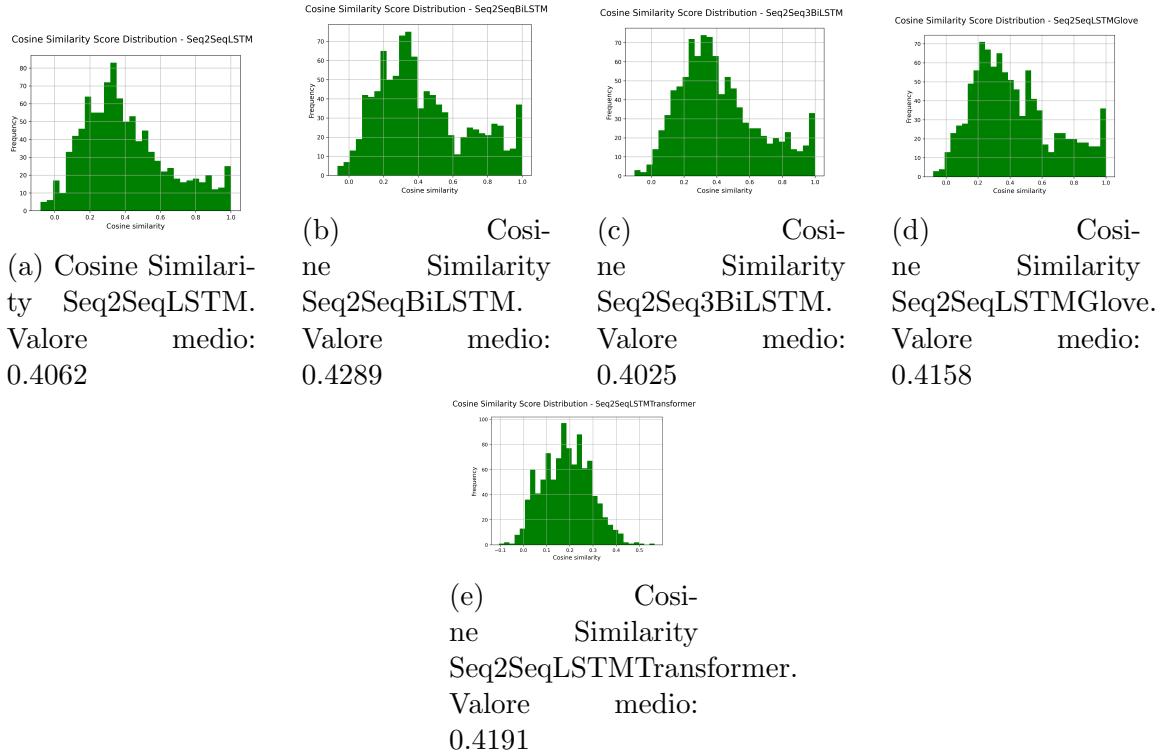


Figura 14: Confronto della cosine similarity tra i modelli Seq2SeqLSTM, Seq2SeqBiLSTM, Seq2Seq3BiLSTM, Seq2SeqLSTMGlove e Seq2SeqLSTMTransformer.

## 7 Conclusioni

Il modello implementato dimostra la capacità di generare riassunti efficaci delle recensioni di prodotti.

I risultati complessivi indicano che il modello Seq2Seq LSTM ha difficoltà significative nel generare output precisi, sia dal punto di vista lessicale che sintattico.

Tuttavia, la similarità cosenica superiore a zero per la maggior parte delle righe suggerisce che il modello riesce a mantenere una correlazione semantica, seppure debole, con il testo di riferimento.

Per un fine di generazione di riassunti ci si può ritenere soddisfatti, grazie alla similarità cosenica.