RELAZIONE:
Text-Summarizer

Enrico Ferraiolo 0001191698

Laurea Magistrale in Informatica

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

Indice

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 sottoinsieme 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 effettutati 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 espanse 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.

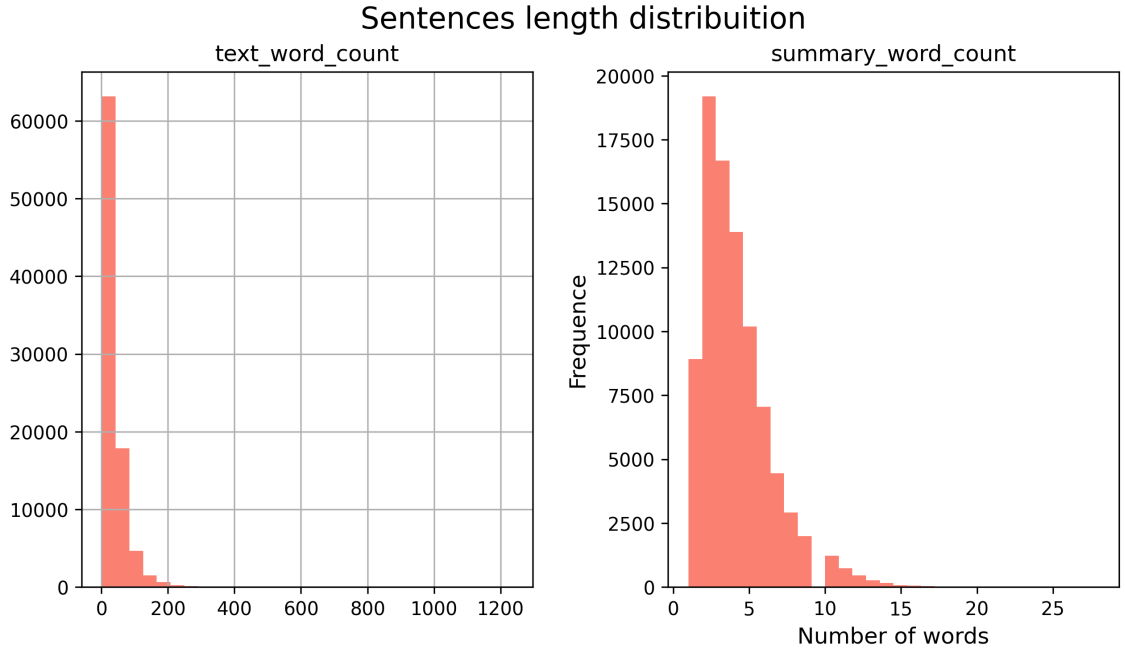

Figura 1: A sinistra la distribuzione delle lunghezze delle recensioni, a destra la distribuzione delle lunghezze dei riassunti

Infatti, come si puè 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 e 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 apprendiento 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

Sono state sperimentante 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, sebbene siano stati effettuati numerosi tentativi e test riportati in seguito in una tabella comparativa.

4.4 Seq2SeqLSTM

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

4.4.1 Training

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

- 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

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 2.

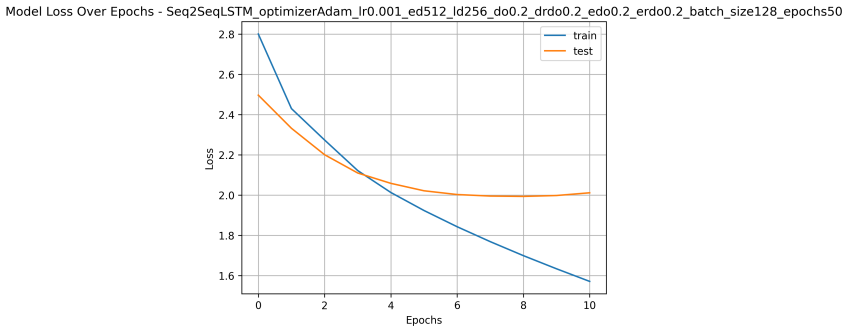

Figura 2: Andamento delle loss durante l'addestramento

4.4.2 Risultati

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

- **Loss**: 1.57
- **Validation loss**: 2.01
- **Accuracy**: 0.67
- **Validation Accuracy**: 0.65

4.4.3 Architettura

L'architettura utilizzata è la seguente:

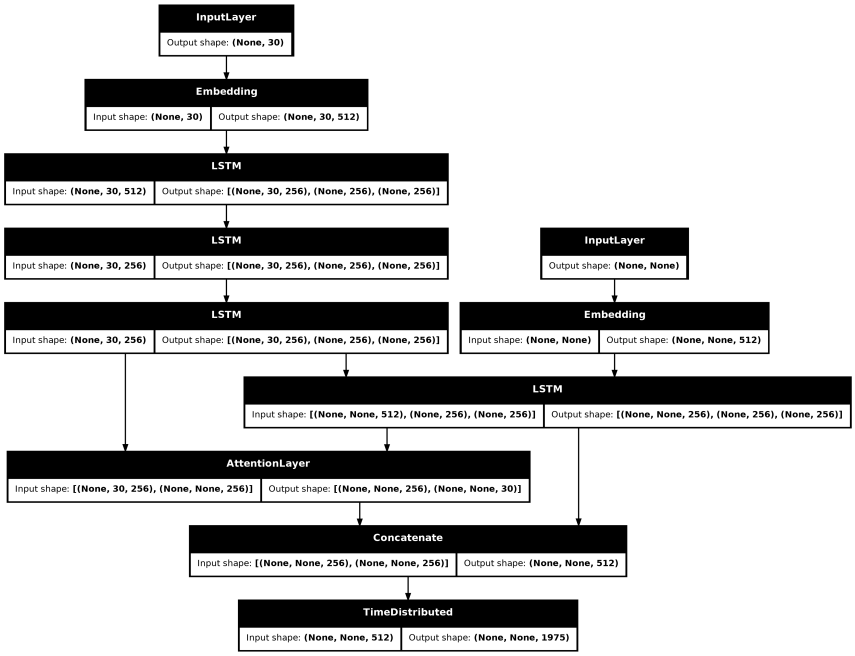

Figura 3: Architettura del modello Seq2SeqLSTM

4.5 Seq2SeqBiLSTM

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

4.5.1 Training

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

- 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: 64
- Epochs: 50

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 4.

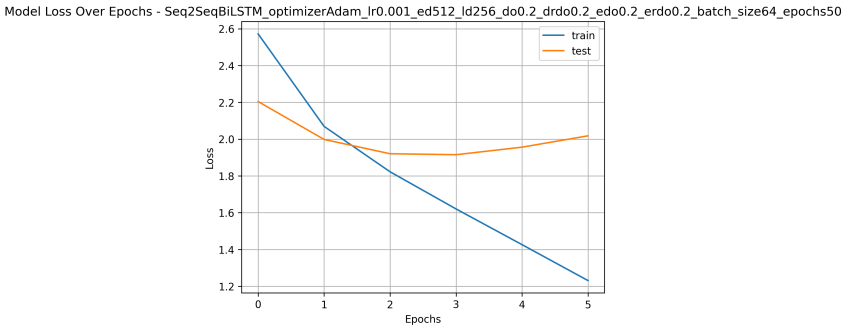

Figura 4: Andamento delle loss durante l'addestramento

4.5.2 Risultati

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

- **Loss**: 1.23
- **Validation loss**: 2.01
- **Accuracy**: 0.72
- **Validation Accuracy**: 0.65

4.5.3 Architettura

L'architettura utilizzata è la seguente:

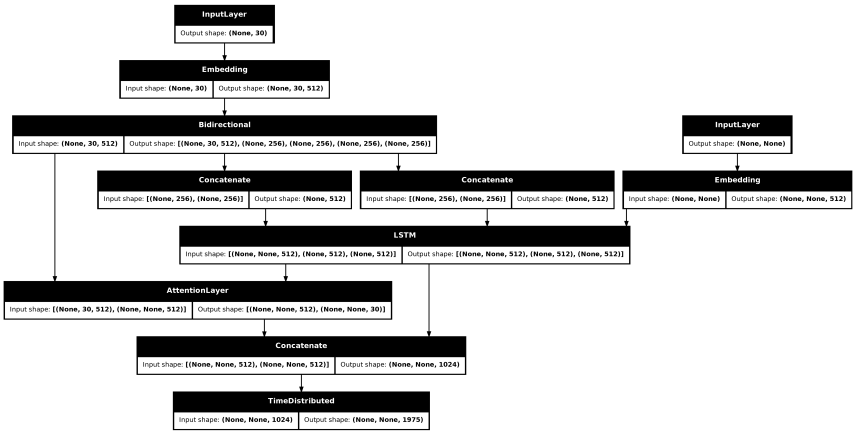

Figura 5: Architettura del modello Seq2SeqBiLSTM

4.6 Seq2Seq3BiLSTM

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

4.6.1 Training

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

- 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: 256
- Epochs: 50

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 6.

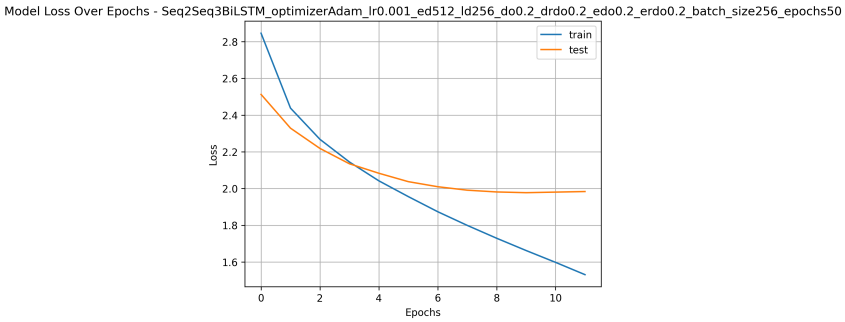

Figura 6: Andamento delle loss durante l'addestramento

4.6.2 Risultati

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

- **Loss**: 1.53
- **Validation loss**: 1.98
- **Accuracy**: 0.68
- **Validation Accuracy**: 0.65

4.6.3 Architettura

L'architettura utilizzata è la seguente:

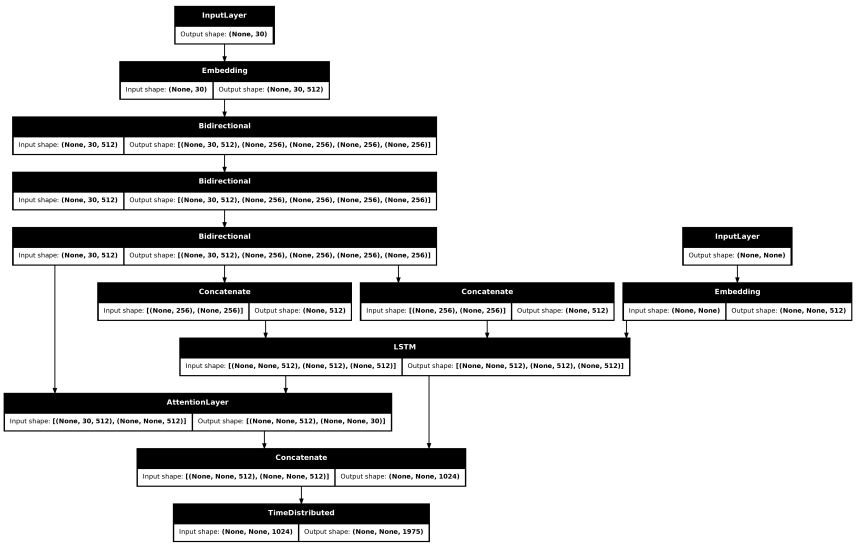

Figura 7: Architettura del modello Seq2Seq3BiLSTM

4.7 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.

4.7.1 Training

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

- 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: 256
- Epochs: 50

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

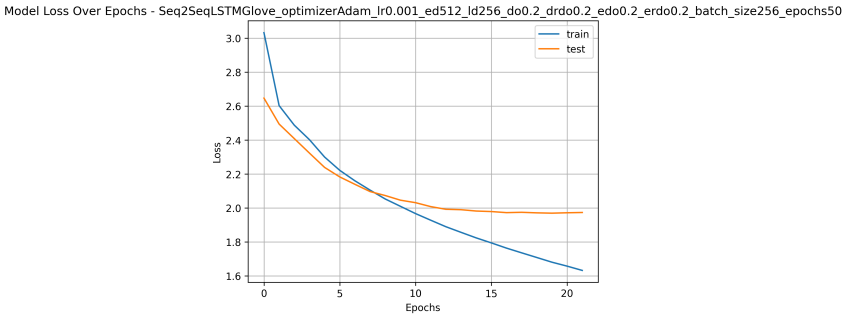

Figura 8: Andamento delle loss durante l'addestramento

4.7.2 Risultati

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

- **Loss**: 1.63
- **Validation loss**: 1.97
- **Accuracy**: 0.66
- **Validation Accuracy**: 0.64

4.7.3 Architettura

L'architettura utilizzata è la seguente:

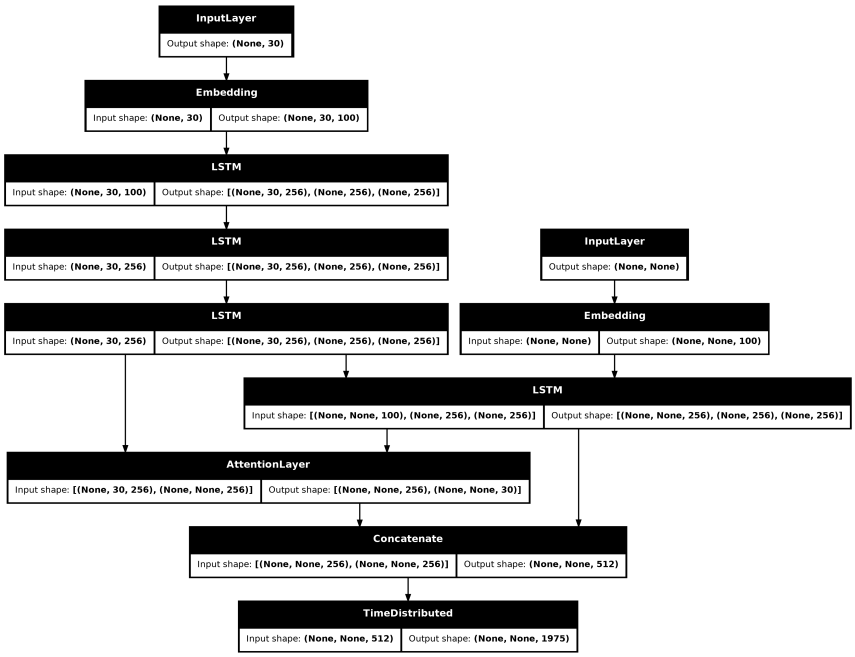

Figura 9: Architettura del modello Seq2SeqLSTMGlove

4.8 Seq2SeqGRU

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

4.8.1 Training

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

- 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: 64
- Epochs: 50

Possiamo verificare l'andamento delle loss durante l'addestramento nella figura 10.

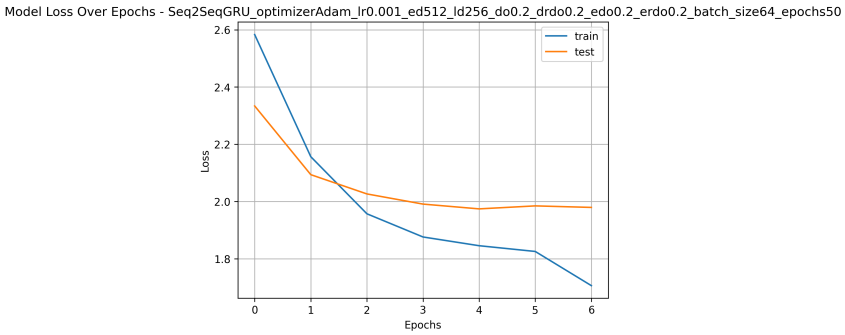

Figura 10: Andamento delle loss durante l'addestramento

4.8.2 Risultati

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

- **Loss**: 1.70
- **Validation loss**: 1.97
- **Accuracy**: 0.66
- **Validation Accuracy**: 0.64

4.8.3 Architettura

L'architettura utilizzata è la seguente:

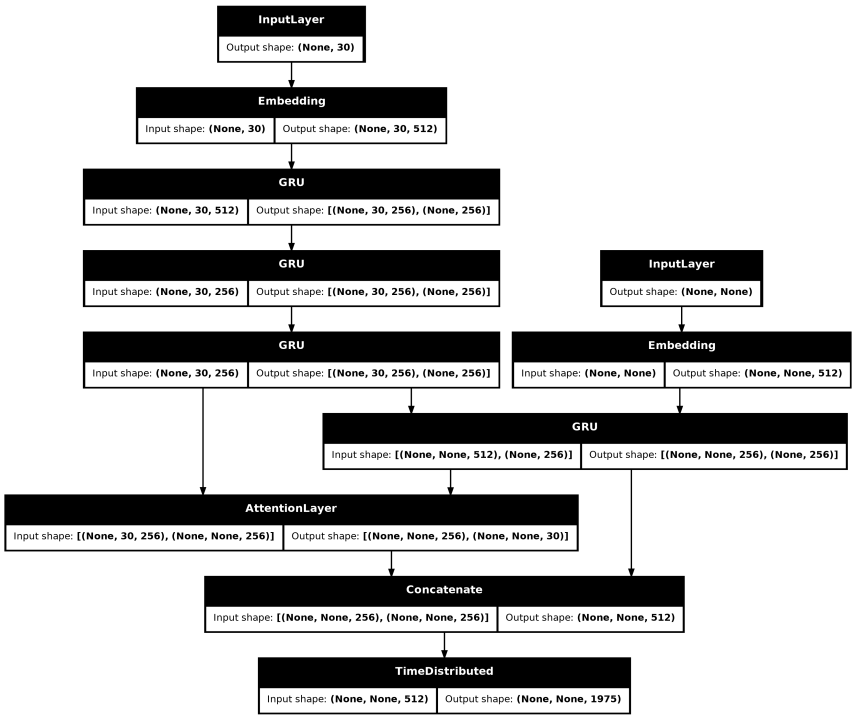

Figura 11: Architettura del modello Seq2SeqGRU

4.9 Confronto tra le Architetture

Model - Instance	mean_cosine	mean_myevaluation	mean_wer	mean_rouge1	mean_rouge2	mean_rougeL
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size128_epochs50_summaries	0.4189	0.4189	1.1440	0.1564	0.0411	0.1552
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size256_epochs50_summaries	0.4318	0.4318	1.1382	0.1665	0.0435	0.1653
Seq2Seq3BiLSTM - Seq2Seq3BiLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size64_epochs50_summaries	0.4198	0.4198	1.1611	0.1588	0.0383	0.1583
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size128_epochs50_summaries	0.4631	0.4631	1.0647	0.2240	0.0696	0.2227
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size256_epochs50_summaries	0.4431	0.4431	1.0895	0.2075	0.0575	0.2052
Seq2SeqBiLSTM - Seq2SeqBiLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size64_epochs50_summaries	**0.4731**	**0.4731**	1.0187	**0.2416**	**0.0738**	**0.2397**
Seq2SeqBiLSTMImproved - Seq2SeqBiLSTMImproved_optimizerAdam_lr0.001_ed300_ld256_do0.3_drdo0.3_edo0.3_erdo0.3_batch_size64_epochs50_summaries	0.1034	0.1034	**1.1710**	0.0136	0.0000	0.0136
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size128_epochs50_summaries	0.4294	0.4294	1.0916	0.1656	0.0417	0.1650
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size256_epochs50_summaries	0.3899	0.3899	1.1170	0.1290	0.0255	0.1287
Seq2SeqGRU - Seq2SeqGRU_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size64_epochs50_summaries	0.4445	0.4445	1.1173	0.1884	0.0563	0.1875
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size128_epochs50_summaries	0.4400	0.4400	1.0826	0.1702	0.0453	0.1694
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size256_epochs50_summaries	0.4374	0.4374	1.0963	0.1629	0.0407	0.1616
Seq2SeqLSTM - Seq2SeqLSTM_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size64_epochs50_summaries	0.4266	0.4266	1.1331	0.1607	0.0454	0.1604
Seq2SeqLSTMGlove - Seq2SeqLSTMGlove_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size128_epochs50_summaries	0.4314	0.4314	1.0877	0.1798	0.0472	0.1795
Seq2SeqLSTMGlove - Seq2SeqLSTMGlove_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size256_epochs50_summaries	0.4467	0.4467	1.0803	0.1927	0.0562	0.1918
Seq2SeqLSTMGlove - Seq2SeqLSTMGlove_optimizerAdam_lr0.001_ed512_ld256_do0.2_drdo0.2_edo0.2_erdo0.2_batch_size64_epochs50_summaries	0.4378	0.4378	1.1131	0.1861	0.0519	0.1855

Figura 12: Comparazione delle istanze dei modelli

5 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.