Indice

Text Summarization

Giuseppe Genito e Rosaria Leone

5 gennaio 2024

1 Introduction

Il presente progetto si propone di sviluppare un algoritmo di *text summarization*, una tecnica che mira a generare riassunti coerenti a partire da testi estesi. Il primo passo cruciale è stato l'implementazione di *web scraping* su una fonte affidabile, in questo caso la rivista online "Il Post". Attraverso questa procedura, siamo riusciti a creare i dataset di addestramento e di test necessari. Successivamente, una volta raccolti i dati, ci siamo dedicati alla fase di pre-processing, che ha coinvolto la pulizia dei testi e la loro tokenizzazione. Questo processo preparatorio è stato fondamentale per garantire che il modello successivamente addestrato possa operare su dati strutturati e coerenti. Il cuore del nostro approccio risiede nell'utilizzo di un modello preaddestrato, in particolare `mBART`. Questa scelta si basa sulla fiducia nella capacità del modello di comprendere e generare riassunti di alta qualità grazie alla sua esperienza pregressa. Tuttavia, non ci siamo limitati all'utilizzo del modello predefinito; al contrario, abbiamo sperimentato variazioni degli iperparametri al fine di ottimizzare le performance del nostro algoritmo.

2 Selezione delle Fonti Online

Il punto di partenza per l'implementazione dell'algoritmo di text summarization è stato la creazione di un dataset di addestramento e test basato su articoli provenienti dalla rivista online *Il Post*. Questa fase è stata realizzata mediante l'utilizzo della libreria BeautifulSoup. Per esplorare e approfondire il codice sorgente, è possibile consultare il repository su GitHub al seguente indirizzo: cliccando qui. I passaggi chiave del codice includono:

- **Funzione di Web Scraping**

  La funzione `coppie(url)` accetta un URL come argomento e restituisce un dizionario contenente i dati estratti.

  - **Introduzione del Ritardo:** Viene introdotto un ritardo di 1 secondo tra le richieste HTTP per rispettare la politica di accesso del sito.
  - **Esecuzione del Web Scraping:** Viene eseguito il web scraping sulla pagina per ottenere gli articoli, in più vengono inizializzate le variabili per contenere gli articoli e i riassunti.
  - **Loop sugli Articoli:** Viene eseguito un loop attraverso gli articoli per estrarre il testo da elementi specifici come `<a>:Titolo` e `<p>:Sottotitolo`, ed infine viene creato un riassunto unendo questi due elementi.
  - **Estrazione degli URL:** Gli URL degli articoli vengono estratti e memorizzati in una lista.
  - **Esecuzione del Loop sugli URL:** Per ciascun articolo, vengono eseguite richieste HTTP separate per ottenere il contenuto della pagina, estraendo il testo dai paragrafi.
  - **Concatenazione dei Paragrafi:** I testi dei paragrafi vengono concatenati e memorizzati in una lista.
  - **Organizzazione dei Dati:** Infine, i dati estratti vengono organizzati in un dizionario contenente due chiavi: 'source' per il testo dell'articolo e 'target' per i riassunti.

Le dimensioni dei data set di train e test creati saranno rispettivamente:

Set	Numero di Articoli
Train	160 articoli
Test	40 articoli

Tabella 1: Train e Test set

3 Preprocessing e Analisi dei Testi

3.1 Data cleaning

I dataset generati sono composti da due colonne principali: una denominata *source*, contenente il testo integrale da cui ottenere il riassunto, e l'altra denominata *target*, contenente i riassunti corrispondenti. Il processo di pulizia dei dati è fondamentale per garantire che il dataset sia di alta qualità e che il modello di text summarization possa apprendere in modo efficace dai dati forniti. In questo paragrafo, esamineremo le fasi di pulizia del dataset, focalizzandoci sulla gestione delle stringhe vuote e dei valori mancanti. Lo scopo è fare in modo che il dataset sia privo di incoerenze o errori che potrebbero compromettere l'efficacia del modello durante la fase di addestramento.

- **Sostituzione di stringhe vuote nella colonna 'source' con NaN** Sostituzione delle stringhe vuote nella colonna 'source' del DataFrame con `np.nan` (valore nullo di NumPy). Lo scopo è trattare le stringhe vuote come valori mancanti.
- **Sostituzione di stringhe vuote nella colonna 'target' con NaN** Stessa operazione di cui sopra, ma per la colonna 'target'.

- **Rimozione di righe con valori mancanti nella colonna 'source'** Rimozione delle righe in cui il valore nella colonna 'source' è mancante (`NaN`). Questa operazione viene eseguita direttamente nel DataFrame.
- **Rimozione delle righe con valori mancanti nella colonna 'target'** Rimozione delle righe in cui il valore nella colonna 'target' è mancante (`NaN`). Anche questa operazione è eseguita direttamente nel DataFrame.
- **Azzeramento dell'indice del DataFrame** L'indice del DataFrame viene resettato quando vengono rimosse le righe, perchè potrebbe avere buchi o potrebbe essere necessario resettarlo per avere indici continui.
- **Stampa le dimensioni delle colonne 'source' e 'target'** L'ultimo step è la stampa delle dimensioni delle colonne 'source' e 'target' dopo le operazioni di pulizia.

3.2 Analisi delle Lunghezze dei Riassunti nel Training Set

Nel seguente istogramma, mostreremo la distribuzione delle lunghezze dei riassunti nel dataset, con un focus particolare su come variano i token nei riassunti associati a IlPost.

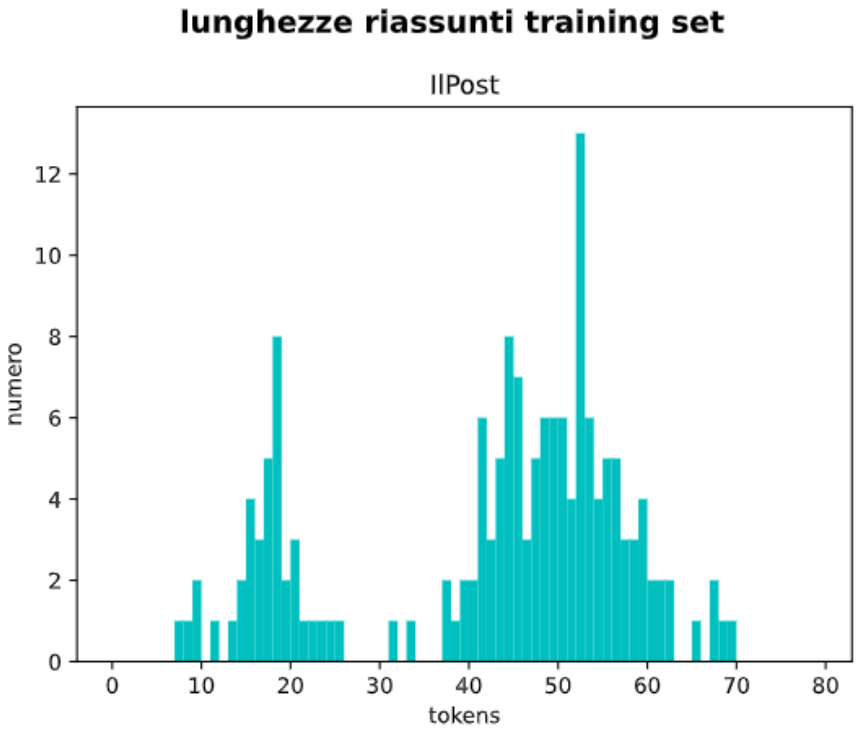

Figura 1: Distribuzione delle lunghezze dei riassunti nel training set.

L'asse delle ascisse rappresenta il numero di token nei riassunti, mentre l'asse delle ordinate indica la frequenza di riassunti corrispondenti a ciascuna lunghezza.

3.3 Analisi delle Lunghezze dei Documenti nel Training Set

Nel seguente istogramma, esploreremo la distribuzione delle lunghezze dei documenti presenti nel nostro training set associato a IlPost.

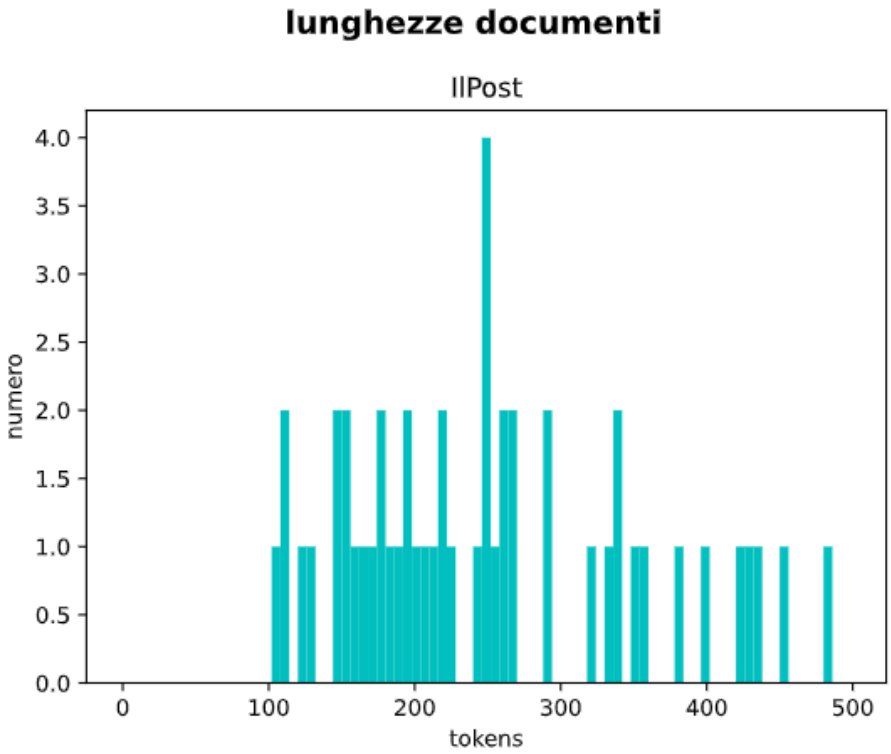

Figura 2: Distribuzione delle lunghezze dei documenti nel training set.

L'asse delle ascisse rappresenta il numero di token nei documenti, mentre l'asse delle ordinate indica la frequenza di documenti corrispondenti a ciascuna lunghezza.

3.4 Pre-processing dei dati

La fase di pre-elaborazione del dataset è un tassello fondamentale nel processo di preparazione dei dati. Di seguito, illustreremo i vari passaggi chiave implementati per garantire la coerenza e la qualità dei dati nel dataset associato a IlPost. Per affrontare eventuali irregolarità nei dati, abbiamo definito una pipeline di pre-processing che consiste in tre passaggi distinti:

```
preproc = preprocessing.make_pipeline(
    preprocessing.normalize.unicode,
    preprocessing.remove.html_tags,
    preprocessing.normalize.whitespace
)
```

- preprocessing.normalize.unicode: Fase in cui vengono normalizzati i caratteri unicode.
- preprocessing.remove.html_tags: Rimozione dei tag HTML.
- preprocessing.normalize.whitespace: Normalizzazione degli spazi bianchi.

Questa fase di pre-elaborazione è essenziale per garantire che i dati siano uniformi, privi di incoerenze e pronti per essere utilizzati nell'addestramento dei modelli di text summarization.

4 Sviluppo del Modello di Text Summarization

Nel corso del nostro progetto di text summarization, abbiamo fatto ampio uso di un modello preaddestrato presentato nel paper di riferimento Landro et al. 2022.

4.1 Modello pre-addestrato

Il modello preaddestrato menzionato è un'implementazione dell'architettura MBart (Multilingual BART), progettata specificamente per la generazione di riassunti in lingua italiana. Questa scelta è stata motivata dalla necessità di adattare il modello alle peculiarità linguistiche della nostra lingua di interesse. Nel codice fornito abbiamo utilizzato:

```
from transformers import MBartTokenizer, MBartForConditionalGeneration
tokenizer = MBartTokenizer.from_pretrained("ARTeLab/mbart-summarization-ilpost")
model = MBartForConditionalGeneration.from_pretrained("ARTeLab/mbart-summarization-ilpost")
```

- la libreria Transformers di *Hugging Face*.
- Poi è stato impiegato il tokenizzatore `MBartTokenizer` per convertire il testo in ingresso in sequenze di token comprensibili per il modello.
- Infine è stato caricato il modello di generazione di riassunti `MBartForConditionalGeneration` tramite il metodo `from_pretrained` con il riferimento al modello specifico `"ARTeLab/mbart-summarization-ilpost"`.

Tra i vantaggi dell'uso di questo modello rientrano:

1. **Specializzazione per il Dominio:** Il modello è stato addestrato specificamente per il dominio di IlPost, migliorando la sensibilità ai tratti unici e al linguaggio specifico di questo contesto.
2. **Multilinguismo:** MBart è progettato per gestire lingue multiple, rendendolo vantaggioso per dataset contenenti testo in diverse lingue.
3. **Generazione Coerente di Testo:** Basato sull'architettura BART, il modello è progettato per generare testo in modo coerente, fondamentale per il compito di text summarization.
4. **Risparmio di Risorse Computazionali:** L'utilizzo di un modello preaddestrato consente di risparmiare risorse computazionali rispetto all'addestramento da zero, migliorando l'efficienza dell'implementazione.
5. **Disponibilità e Facilità d'Uso:** I modelli preaddestrati, come quelli forniti da Hugging Face Transformers, sono facilmente accessibili e possono essere integrati rapidamente nei progetti, semplificando il processo di sviluppo.

4.2 Fase di addestramento

Di seguito verranno illustratii i vari step della fase di addestramento:

Tokenizzazione dei dati: La tokenizzazione è una fase cruciale nel pre-processing dei dati per l'addestramento di modelli di text summarization. I modelli di machine learning, inclusi quelli per la generazione di testo, operano su sequenze di token anziché su testo grezzo. Tokenizzare i dati di addestramento crea rappresentazioni numeriche dei testi che possono essere utilizzate dal modello per apprendere i pattern durante l'addestramento. Inoltre, l'uso di tensori PyTorch facilita il processo di addestramento utilizzando il framework PyTorch. L'aggiunta di padding e la troncatura assicurano che tutti i dati di addestramento abbiano dimensioni compatibili, un requisito comune per l'addestramento efficiente di modelli di deep learning.

```
train_inputs = tokenizer(train_texts, return_tensors="pt", padding=True, truncation=True)
```

- **tokenizer** è l'oggetto che rappresenta il tokenizzatore utilizzato. In questo contesto, è un tokenizzatore associato ad un modello di generazione di riassunti.

- **train_texts** rappresenta i testi di input del set di dati di addestramento.
- **return_tensors="pt"** specifica di restituire i dati tokenizzati sotto forma di tensori PyTorch.
- **padding=True** indica di aggiungere il padding ai dati tokenizzati in modo che abbiano la stessa lunghezza.
- **truncation=True** significa che i testi saranno troncati se superano la lunghezza massima consentita dal tokenizzatore.

I dataset PyTorch sono strutturati per essere compatibili con il framework PyTorch e semplificare il processo di addestramento di modelli di machine learning. Creare dataset PyTorch è essenziale per organizzare i dati in modo che possano essere facilmente utilizzati durante la fase di addestramento. Utilizzando TensorDataset, è possibile mantenere la correlazione tra gli input e le etichette, consentendo al modello di apprendere la relazione tra i due durante l'addestramento. Questi dataset saranno successivamente utilizzati con un DataLoader PyTorch per creare batch di dati durante l'iterazione attraverso le epoche di addestramento e test. Questa struttura dati facilita il passaggio dei dati al modello durante l'addestramento e semplifica il processo di valutazione del modello sui dati di test.

```
train_dataset = TensorDataset(train_inputs["input_ids"], train_labels["input_ids"])
```

- **train_inputs["input_ids"]** rappresenta i tensori contenenti gli ID dei token degli input del dataset di addestramento.
- **train_labels["input_ids"]** rappresenta i tensori contenenti gli ID dei token delle etichette (riassunti) del dataset di addestramento.
- **TensorDataset** crea un dataset PyTorch che può essere utilizzato per iterare attraverso batch di input e target durante l'addestramento del modello.

La classe **DataLoader** di PyTorch crea un iteratore che può essere utilizzato per iterare attraverso il dataset di addestramento in batch durante il processo di addestramento del modello.

```
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
```

- **batch_size**: Specifica il numero di campioni (istanze) che devono essere processati in ogni iterazione. L'addestramento di modelli di deep learning è spesso eseguito su batch di dati invece che su un singolo esempio alla volta. batch_size definisce la dimensione di ciascun batch.
- **shuffle=True**: Indica che i dati devono essere mescolati prima di essere suddivisi in batch. Il mescolamento è utile per evitare che il modello apprenda a dipendere dall'ordine specifico dei dati di addestramento e per introdurre casualità nel processo di apprendimento.

L'ottimizzatore è fondamentale per l'addestramento di modelli di machine learning. Durante l'addestramento, il modello cerca di ridurre una funzione di perdita, che rappresenta l'errore tra le sue predizioni e gli obiettivi desiderati (le etichette del dataset). L'ottimizzatore è responsabile di aggiornare i parametri del modello in modo che questa perdita venga minimizzata.
Adam è scelto per la sua efficacia in una vasta gamma di scenari di addestramento. Il tasso di apprendimento, specificato come lr, controlla quanto velocemente l'ottimizzatore si muove nella direzione opposta al gradiente della funzione di perdita. La scelta del tasso di apprendimento è critica e può influenzare la convergenza e le prestazioni del modello.

```
optimizer = torch.optim.Adam(model.parameters(), lr=1e-6)
```

- **torch.optim.Adam**: Adam è un algoritmo di ottimizzazione ampiamente utilizzato nell'addestramento di modelli di machine learning. Ottimizzatori come Adam regolano i pesi del modello durante l'addestramento per ridurre la perdita, ovvero l'errore tra le predizioni del modello e gli obiettivi desiderati.
- **model.parameters()**: Questa funzione fornisce tutti i parametri del tuo modello che devono essere ottimizzati durante l'addestramento. Gli ottimizzatori regolano questi parametri per minimizzare la funzione di perdita.

- **lr=1e-6**: Questo parametro imposta il tasso di apprendimento (learning rate) dell'ottimizzatore. Il tasso di apprendimento controlla la dimensione dei passi che l'ottimizzatore compie durante l'aggiornamento dei pesi del modello. Un tasso di apprendimento troppo alto può causare oscillazioni o la mancata convergenza, mentre un tasso di apprendimento troppo basso può rallentare l'addestramento.

Infine **torch.nn.CrossEntropyLoss()** è una funzione di perdita (loss function) spesso utilizzata in combinazione con l'ottimizzatore Adam (o altri) durante l'addestramento di reti neurali per problemi di classificazione multiclasse. È una scelta comune perché è adatta per problemi in cui ogni esempio appartiene a una sola classe. Ognuna di queste linee di codice svolge un ruolo specifico nell'implementazione e nell'addestramento di un modello di text summarization.

5 Ottimizzazione e Valutazione

Gli iperpapametri che abbiamo variato durante i vari addestramenti con lo scopo di ottimizzare le performace sono i seguenti:

Numero di Epoche

- **Descrizione:** Le epoche rappresentano il numero di volte che l'intero set di dati di addestramento viene presentato al modello durante il processo di addestramento.
- **Variabilità:** Tipicamente, il numero di epoche può variare da poche (ad esempio, 5) a decine o anche centinaia, a seconda delle dimensioni del set di dati e della complessità del modello. Un valore basso può portare a sottoutilizzazione (underfitting), mentre un valore troppo alto può portare a sovrautilizzazione (overfitting).

Batch Size

- **Descrizione:** Il batch size rappresenta il numero di campioni di addestramento utilizzati in una singola iterazione durante l'aggiornamento dei pesi del modello.
- **Variabilità:** Può variare da valori bassi come 4 o 8 a valori più alti come 32, 64, o anche 128. Valori troppo alti possono richiedere più memoria, mentre valori troppo bassi potrebbero rallentare l'addestramento.

Tasso di Apprendimento (Learning Rate)

- **Descrizione:** Il tasso di apprendimento controlla la dimensione dei passi fatti durante l'aggiornamento dei pesi del modello. È una scala per la correzione delle previsioni del modello in base all'errore rispetto alle etichette.
- **Variabilità:** Può variare da valori bassi come 1×10^{-5} o 1×10^{-4} a valori più alti come 1×10^{-2} o 1×10^{-3}. Valori troppo alti possono causare oscillazioni o mancata convergenza, mentre valori troppo bassi possono rallentare l'addestramento.

5.1 Migliore configurazione

Dopo molti tentativi la configurazione migliore è stata la seguente:

Numero di epoche	Batch size	learning rate
10	2	1e-6

Tabella 2: Esempio di tabella con due righe e tre colonne.

L'andamento della loss durante l'addestramento è stato il seguente:

```
Epoch 1/10, Average Loss: 7.2756
Epoch 2/10, Average Loss: 6.6440
Epoch 3/10, Average Loss: 6.2767
Epoch 4/10, Average Loss: 5.9695
Epoch 5/10, Average Loss: 5.6936
Epoch 6/10, Average Loss: 5.4166
Epoch 7/10, Average Loss: 5.1553
Epoch 8/10, Average Loss: 4.9042
Epoch 9/10, Average Loss: 4.6647
Epoch 10/10, Average Loss: 4.4035
```

Figura 3: Valori della loss

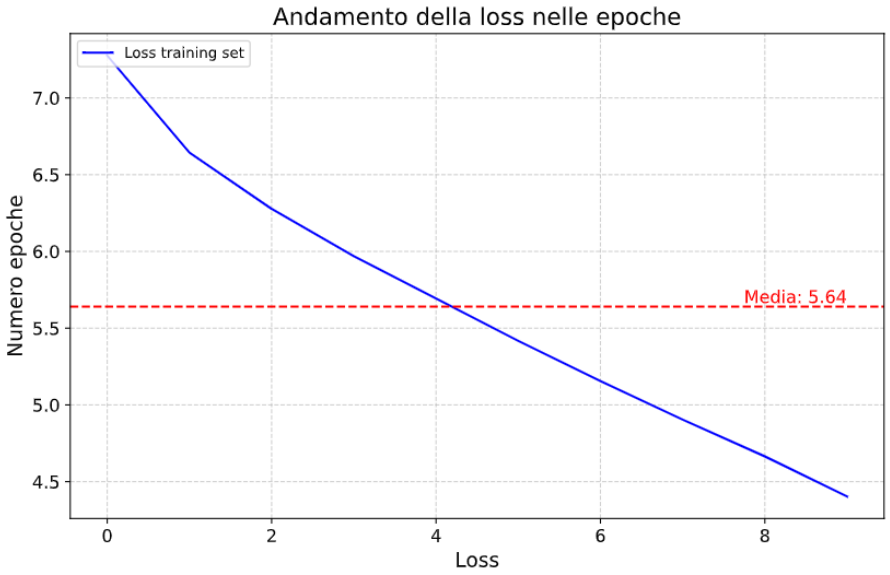

Figura 4: Andamento della loss

La funzione di perdita è una misura dell'errore tra le previsioni del modello e i valori effettivi del dataset di addestramento. Come è possibile osservare dal grafico l'andamento della loss è descrescente questo vuol dire che il modello sta migliorando nella sua capacità di adattarsi ai dati di addestramento.

5.2 Misure di accuratezza

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

- **Significato:** ROUGE è una serie di metriche di valutazione automatica progettate per misurare la qualità di un riassunto rispetto ai riferimenti umani, concentrandosi sulla sovrapposizione di parole e frasi.
- **Metriche Comuni ROUGE:**
  - ROUGE-N (N-gram overlap): Misura la sovrapposizione di N-grammi tra il riassunto e il riferimento.
  - ROUGE-L (Longest Common Subsequence): Misura la lunghezza della sottosequenza comune più lunga tra il riassunto e il riferimento.
  - ROUGE-W (Weighted N-gram overlap): Variante di ROUGE-N con pesi diversi per N-grammi di lunghezze diverse.

- **Intervalli di Valori:** Da 0 a 1, dove 0 indica nessuna sovrapposizione e 1 indica corrispondenza perfetta.

BLEU (Bilingual Evaluation Understudy)

- **Significato:** BLEU è una metrica di valutazione automatica per misurare la qualità delle traduzioni automatiche, adattabile per valutare la qualità dei riassunti.
- **Metrica Comune BLEU:**
  - BLEU Score: Calcola la precisione degli N-grammi del testo generato rispetto ai riferimenti umani.
- **Intervalli di Valori:** Da 0 a 1, con spesso espressione in percentuale (0% - 100%).

Considerazioni Generali

- Entrambe le metriche forniscono indicazioni sulla qualità del testo generato rispetto ai riferimenti umani.
- Valori più alti indicano una maggiore similarità tra il testo generato e i riferimenti umani.
- Le metriche ROUGE e BLEU sono complementari e spesso utilizzate insieme per una valutazione più completa delle prestazioni del modello di text summarization.

Nel contesto dell'analisi delle metriche ROUGE e BLUE abbiamo ottenuto i seguenti risultati::

```
ROUGE Scores:
ROUGE-1: {'r': 0.34788617074706885, 'p': 0.28812183322362606, 'f': 0.3095453808233408}
ROUGE-2: {'r': 0.16273052941538357, 'p': 0.13620381937538836, 'f': 0.14574129974315536}
ROUGE-L: {'r': 0.3074594848136582, 'p': 0.2554170893946396, 'f': 0.2740756700090432}
```

Figura 5: ROUGE

- **ROUGE-1: 0.35**: Questo valore indica che circa il 35% degli unigrammi nel riassunto generato coincide con quelli presenti nei riferimenti.
- **ROUGE-2: 0.16**: Il valore di 0.16 suggerisce che circa il 16% dei bigrammi nel riassunto generato coincide con quelli presenti nel campione di confronto.
- **ROUGE-L: 0.31**: La metrica ROUGE-L, che considera l'overlap più lungo tra il sistema e il riferimento, ha prodotto un valore di 0.31, indicando che circa il 31% delle parole più lunghe è in comune tra il riassunto generato e il testo di riferimento.

Questi risultati forniscono una valutazione della corrispondenza tra il sistema di text summarization e il set di test, rappresentando un'indicazione della qualità complessiva del riassunto generato.

```
BLEU Score (intero corpus): 0.0604199308981629
```

Figura 6: BLUE

Questa metrica indica che il 6% delle parole o frasi nella traduzione coincide con quelle nel test set.

6 Conclusioni

Considerando le limitate risorse computazionali a disposizione, abbiamo esplorato diverse configurazioni di iperparametri con un numero limitato di tentativi. Nonostante questa restrizione, siamo soddisfatti dei risultati ottenuti, poiché l'algoritmo implementato dimostra la capacità di generare riassunti esaustivi partendo da un testo di input. La nostra esperienza limitata nell'ottimizzazione degli iperparametri non ha impedito il raggiungimento di risultati promettenti, suggerendo il potenziale del nostro approccio nella generazione di riassunti informativi.

Riferimenti bibliografici

[1] Nicola Landro et al. *Due nuovi dataset per il riassunto di testi astratti in lingua italiana*. Presentazione ricevuta: 24 marzo 2022 / Revisione: 23 aprile 2022 / Accettato: 27 aprile 2022 / Pubblicato: 29 Aprile 2022. 2022. URL: https://www.mdpi.com/2078-2489/13/5/228.

[2] Banda Larga. *ILAS: Summarization with T5 and Mbart*. anno.