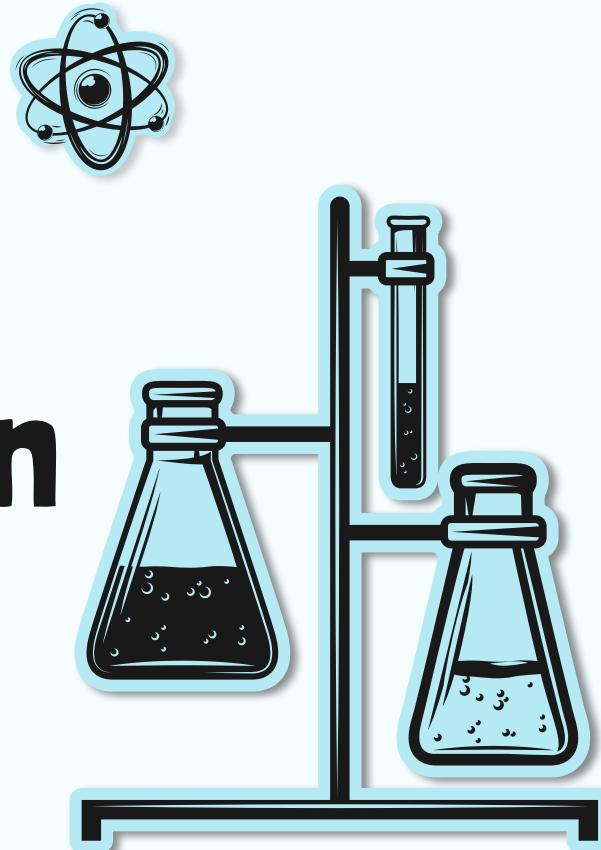


Text summarization

Sistema per sintetizzare testi in italiano



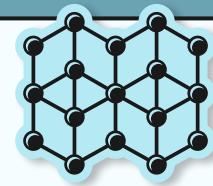
Contenuti del progetto



Indice delle sfide affrontate per la realizzazione del progetto

| | |
|--|--|
| <u>Scelta delle fonti</u> | Scegliere quali fonti utilizzare tra articoli, blog e wiki |
| <u>Scraping</u> | Implementazione di un algoritmo per estrarre informazioni dalle pagine web |
| <u>Preprocessing</u> | Pulizia, formattazione e analisi del testo |
| <u>Addestramento modello pre addestrato</u> | Addestramento del modello pre-addestrato BART sui nuovi dati |
| <u>Valutazione con ROUGE e BLUE</u> | Valutazione delle performance su dati di test con ROUGE e BLUE |
| <u>Utilizzo reale</u> | Dimostrazione di un utilizzo reale del modello |

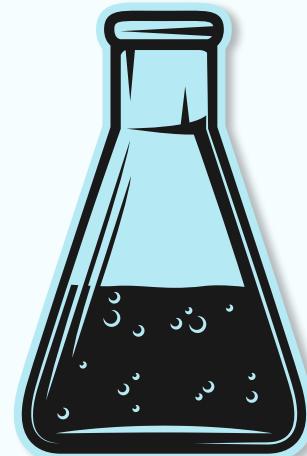




01

Scelta delle fonti

Scelta della fonte per la creazione del dataset



Scelta delle fonti

01

Il Post

Titoli e sottotitoli molto esplicativi, testi lunghi, url accessibili

[Tecnologia - Il Post](#)

03

Wikipedia

Impossibilità di sintetizzare il testo.

02

Fan Page

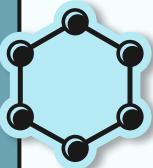
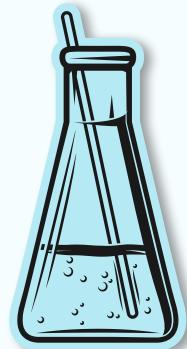
Caotico, difficile reperibilità del testo.

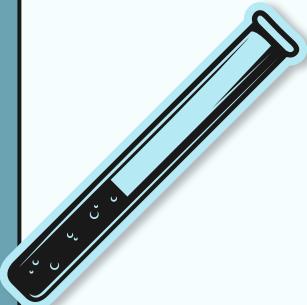
Titoli allarmistici, e clickbait

04

ANSA

Mancanza del titolo secondario

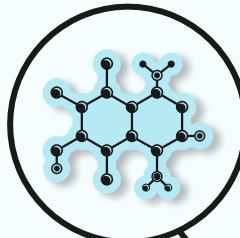




Il Post!

Il post dispone in ogni pagina
sempre 20 articoli, tutti reperibili
tramite il codice:

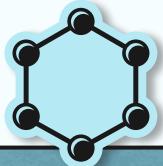
```
articles = soup.find_all("article",
class_ = "taxonomy-item_q6jgq_1 _opener_q6jgq_14")
```



Perché estrarre dalle pagine Web

Motivo 1

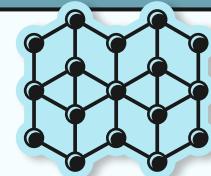
Attraverso l'unione del titolo e del titolo secondario si cattura precisamente l'essenza del concetto



Motivo 2

La quantità di articoli presenti è elevata. Permettendo di creare dataset lunghi.





02

Scraping

Estrazione delle informazioni dagli articoli in modo automatico



Fattori importanti per lo scraping

I componenti che bisogna estrarre sono:

- Pagina web dell'articolo
- Titolo e sottotitolo
- Paragrafi del testo

Ogni pagina contiene 20 articoli in ordine temporale, scandagliati come segue:

- “<https://www.ilpost.it/tecnologia/>” + “page/” + numero/
- Link 1: <https://www.ilpost.it/tecnologia/>
- Link 2: <https://www.ilpost.it/tecnologia/> —— page/2/



Divisione degli elementi

URL pagine articoli

Costruzione dell'url per raggiungere le pagine degli articoli



Selezione degli articoli

Selezione dei 20 articoli per ogni pagina, come mostrato sopra

Selezione titolo e sottotitolo

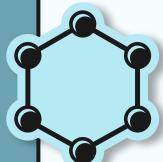
La selezione del titolo e del sottotitolo avviene nella lista degli articoli, prima della loro apertura

Selezione del testo

Cattura del link dell'articolo e selezione del testo della pagina



Selezione degli articoli



10/11/2023

Negli Stati Uniti Apple pagherà una multa di 25 milioni di dollari per aver favorito l'assunzione di lavoratori immigrati rispetto a cittadini statunitensi



10/11/2023

Meta ora vuole concentrarsi su WhatsApp

Mark Zuckerberg pensa che sia il contenitore più adatto a diventare il social del futuro, e forse un'app in cui fare un po' di tutto



6/11/2023

Inizia il processo di Epic Games contro Google

Il produttore del videogioco Fortnite accusa Google di esercitare un monopolio nella distribuzione e nei pagamenti



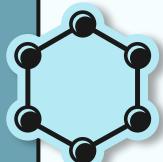
```
<div class="row index_row-wrap__aFB00 index_row-two-cols__Uv030">
  flex
    <div class="col col-xl-9 index_home-left__ikJqd">
      article class="_taxonomy-item_q6jgq_1 _opener_q6jgq_14">
        flex
          <time class="_taxonomy-item_time_q6jgq_37 col-lg-1 col-sm-1 2">10/11/2023</time>
          ...
        <div class="_taxonomy-item_content_q6jgq_53 col-lg-7 col-8">
          $0
        <a href="https://www.ilpost.it/2023/11/10/apple-multa-assunzioni-immigrati/">
          h2 class="_article-title_1aaqi_4" style="font-size:22px; font-weight:700">...</h2>
          p class="_article-paragraph_e98aq_1" style="font-size:16px;line-height:18px;font-weight:400;color:#707070">...</p>
        </a>
      </div>
      figure class="_taxonomy-item_image_q6jgq_72 col-lg-3 col-3 ...</figure>
    </article>
    <article class="_taxonomy-item_q6jgq_1 _opener_q6jgq_14">
      ...
    </article>
  </div>
  div._taxonomy-item_content_q6jgq_53.col-lg-7.col-8

```

Stili Calcolati Layout Listener di eventi Punti di interruzione DOM



Selezione degli articoli



10/11/2023

Negli Stati Uniti Apple pagherà una multa di 25 milioni di dollari per aver favorito l'assunzione di lavoratori immigrati rispetto a cittadini statunitensi



10/11/2023

Meta ora vuole concentrarsi su WhatsApp

Mark Zuckerberg pensa che sia il contenitore più adatto a diventare il social del futuro, e forse un'app in cui fare un po' di tutto



6/11/2023

Inizia il processo di Epic Games contro Google

Il produttore del videogioco Fortnite accusa Google di esercitare un monopolio nella distribuzione e nei pagamenti delle app su Android



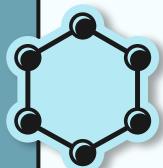
```
i > ... </div> (flex)
▼ <div class="row index_row-wrap__aFB00 index_row-two-cols_Uv030" > (flex)
  ▼ <div class="col col-xl-9 index_home-left__ikJqd" >
    ▼ <article class="_taxonomy-item_q6jgq_1 _opener_q6jgq_14" > (flex)
      <time class="_taxonomy-item_time_q6jgq_37 col-lg-1 col-sm-12" >10/11/2023</time> (flex)
      ▼ <div class="_taxonomy-item_content_q6jgq_53 col-lg-7 col-8" >
        ... (flex)
        ▲ <a href="https://www.ilpost.it/2023/11/10/apple-multa-assunzioni-immigrati/" > == $0
          ▶ <h2 class="_article_title_1aaqi_4" style="font-size:22px;font-weight:700">...</h2>
          <p class="_article-paragraph_e98aq_1" style="font-size:16px;line-height:18px;font-weight:400;color:#707070"></p>
        </a>
        </div>
        ▶ <figure class="_taxonomy-item_image_q6jgq_72 col-1" > (flex)
          ... (flex)
        </figure>
      </div>
    </article>
    <div class="index_home-right__kzLWU" >
      ... (flex)
    </div>
  </div>
  <div class="col col-xl-3 index_home-right__kzLWU" >
    ... (flex)
  </div>
</div>
```

Stili Calcolati Layout Listener di eventi

Filtro :hov .cls + ↗ ↘



Selezione titolo e sottotitolo



Selezione del titolo attraverso il tag `<a>`

10/11/2023

Meta ora vuole concentrarsi su WhatsApp

Mark Zuckerberg pensa che sia il contenitore più adatto a diventare il social del futuro, e forse un'app in cui fare un po' di tutto



```
-12'>10/11/2023</time> flex
  <div class="__ taxonomy-item__ content_q6jgg_53 col-lg-7 col-8">
    <a href="https://www.ilpost.it/2023/11/10/whatsapp-meta-piani/"> == $0
      <h2 class="__ article-title_1aaqi_4" style="font-size:22px;font-weight:700">Meta ora vuole concentrarsi su WhatsApp</h2>
      <p class="__ article-paragraph_e98aq_1" style="font-size:16px;line-height:18px;font-weight:400;color:#707070">...</p>
    </a>
  </div>
  <figure class="__ taxonomy-item__ image_q6jgg_72 col-lg-3 col-3"> ... </figure>
</article>
```

Selezione del sottotitolo attraverso il tag `<p>`



Codice scraping

```
r = requests.get(url)
soup = BeautifulSoup(r.text, "html5lib")
articles = soup.find_all("article", class_="_taxonomy-item_q6jqa_1 _opener_qs6jqa_14")
# list of summaries
summary = []
for i in range(len(articles)):
    try:
        testo1 = articles[i].select_one("a").get_text().strip()
    except:
        testo1 = ""
    try:
        testo2 = articles[i].select_one("p").get_text().strip()
    except:
        testo2 = ""

    if testo1[-1] == "?":
        if testo2 == ".": riassunto = testo1
        elif testo2[-1] == ".": riassunto = testo1 + " " + testo2[1:]
        else: riassunto = testo1 + " " + testo2[0].lower() + testo2[1:] + "."
    elif testo1[-1] == ":":
        if testo2 == ".": riassunto = testo1[:-1] + "."
        elif testo2[-1] == ".": riassunto = testo1 + " " + testo2[0].lower() + testo2[1:]
        else: riassunto = testo1 + " " + testo2[0].lower() + testo2[1:] + "."
    elif testo1[-1] != ".": 
        if testo2 == ".": riassunto = testo1 + "."
        elif testo2[-1] == ".": riassunto = testo1 + " " + testo2
        else: riassunto = testo1 + " " + testo2 + "."

    summary.append((riassunto))

# articles urls
pagelinks = []
print("Lunghezza articoli: ", len(articles))

for i in range(len(articles)):
    url = articles[i].find_all('a')[0]
    pagelinks.append(url.get('href'))

thearticle = []
paragraph_test = []
print("Lunghezza pagelinks: ", len(pagelinks))
for link in pagelinks:
    paragraph_text = []
    url = link
    page = requests.get(url)
    time.sleep(0.2)
    soup = BeautifulSoup(page.text, 'html.parser')

    testo_articolo = soup.find_all("p")
    for paragraph in testo_articolo[:-1]:
        text = paragraph.get_text()
        paragraph_text.append(text)

    testo_fin = ' '.join(paragraph_text)
    thearticle.append(testo_fin)
    d = {'source': thearticle, 'target': summary}
    return d
```

Selezione articoli

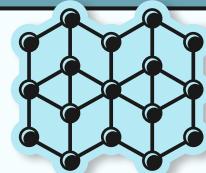
Cattura titolo e sottotitolo

Unione del titolo e del sottotitolo

Salvataggio dei link degli articoli

Apertura URL salvati

Cattura del testo dell'articolo



03

Preprocessing

Pulizia del testo formattazione e analisi



Punti affrontati

Unione di titolo e sottotitolo

Aggiunta di punteggiatura, spaziatura e altro.

Rimozione testi vuoti ed NA

Rimozioni di errori nella fase di scraping

Textacy

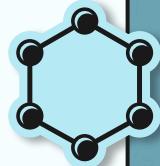
Utilizzo di textacy per il cleaning, normalizzazione unicode e rimozione tag

Analisi lunghezza testi

Analisi grafica della lunghezza dei testi di sintesi e dei paragrafi

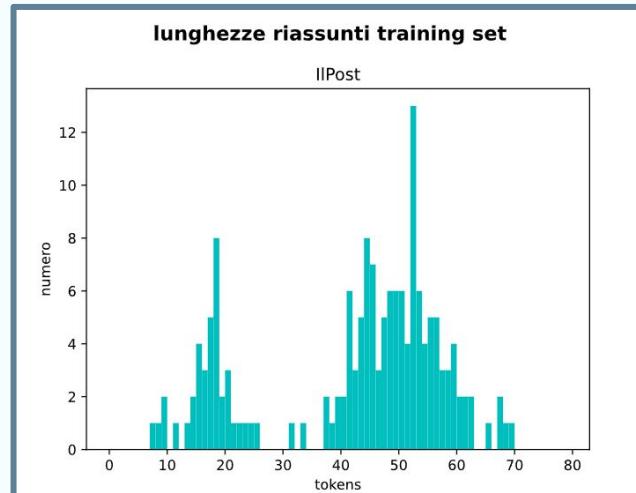
Tokenizzazione

Tokenizzazione dei testi, sia sintesi che paragrafi e sia per il training che per il test set, attraverso il comando tokenizer.

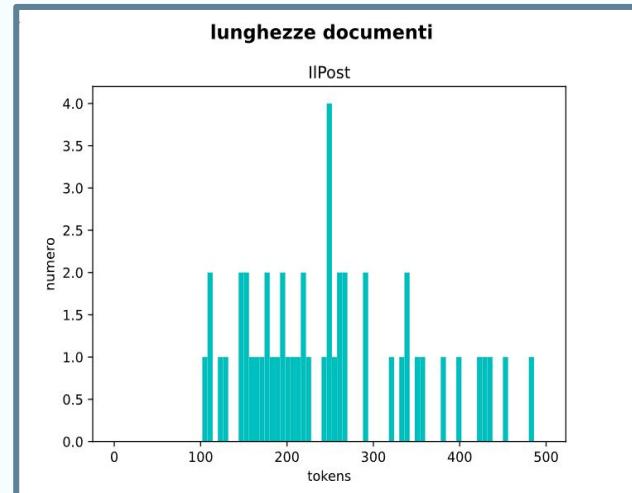


Analisi grafica lunghezza testi

Lunghezza riassunti



Lunghezza riassunti

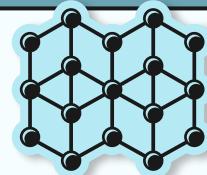


Sull'asse x il numero di parole nel testo

04

Modello MBART

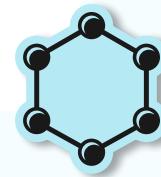
Utilizzo di un modello pre-addestrato per la sintesi del testo



Import del modello pre-addestrato

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

| | | |
|--------------------------|------|-------------------------------------|
| tokenizer_config.json: | 100% | 474/474 [00:00<00:00, 38.3kB/s] |
| sentencepiece.bpe.model: | 100% | 5.07M/5.07M [00:01<00:00, 2.59MB/s] |
| special_tokens_map.json: | 100% | 495/495 [00:00<00:00, 42.9kB/s] |
| tokenizer.json: | 100% | 9.08M/9.08M [00:01<00:00, 5.26MB/s] |
| config.json: | 100% | 1.43k/1.43k [00:00<00:00, 110kB/s] |
| model.safetensors: | 100% | 2.44G/2.44G [08:40<00:00, 3.01MB/s] |



Dal sito Hugging Face abbiamo preso il codice per utilizzare il modello

Link alla pagina: [ARTeLab/mbart-summarization-ilpost · Hugging Face](https://huggingface.co/ARTeLab/mbart-summarization-ilpost)



```
# Tokenizzazione dei dati  
  
train_inputs = tokenizer(train_texts, return_tensors="pt", padding=True, truncation=True)  
train_labels = tokenizer(train_targets, return_tensors="pt", padding=True, truncation=True)  
  
test_inputs = tokenizer(test_texts, return_tensors="pt", padding=True, truncation=True)  
test_labels = tokenizer(test_targets, return_tensors="pt", padding=True, truncation=True)
```

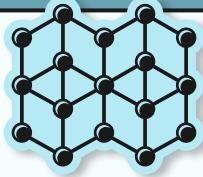
```
# Crea il dataset PyTorch  
train_dataset = TensorDataset(train_inputs["input_ids"], train_labels["input_ids"])  
test_dataset = TensorDataset(test_inputs["input_ids"], test_labels["input_ids"])  
# Crea il dataloader per il training  
batch_size = 2  
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)  
test_dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=True)
```

Tokenizzazione

return_tensors=“pt” da l’output sotto forma di tensori Pytorch. Padding aggiunge del padding ai token in modo che abbiano la stessa lunghezza. Truncation, tronca i token che sono troppo lunghi

Dataset pytorch e dataloader

I dataset tokenizzati sono utilizzati per creare dei dataset PyTorch, i DataLoader invece sono responsabili della divisione dei dati in batch



Train del modello



```
# Addestra il modello
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    total_loss = 0.0
    for batch in tqdm(train_dataloader, desc=f'Epoch {epoch + 1}/{num_epochs}', leave=False):
        inputs = batch[0]
        labels = batch[1]

        optimizer.zero_grad()
        outputs = model(inputs, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    average_loss = total_loss / len(train_dataloader)
    loss_media_train.append(average_loss)
    print(f'Epoch {epoch + 1}/{num_epochs}, Average Loss: {average_loss:.4f}')
```



Valutazione sul test



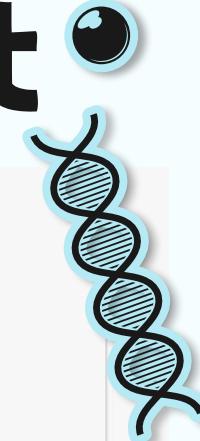
```
model.eval()
predictions = []
references = []

with torch.no_grad():
    for batch in test_dataloader:
        inputs = batch[0]
        labels = batch[1]

        outputs = model.generate(inputs)

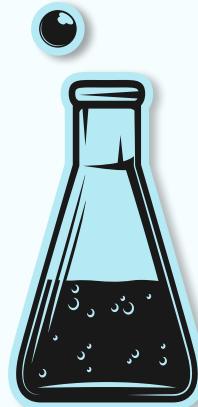
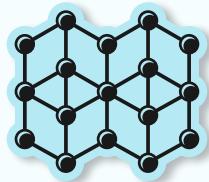
        # Decodifica le predizioni e i riferimenti
        predicted_texts = tokenizer.batch_decode(outputs, skip_special_tokens=True)
        target_texts = tokenizer.batch_decode(labels, skip_special_tokens=True)

        # Aggiungi le predizioni e i riferimenti alle liste
        predictions.extend(predicted_texts)
        references.extend(target_texts)
```

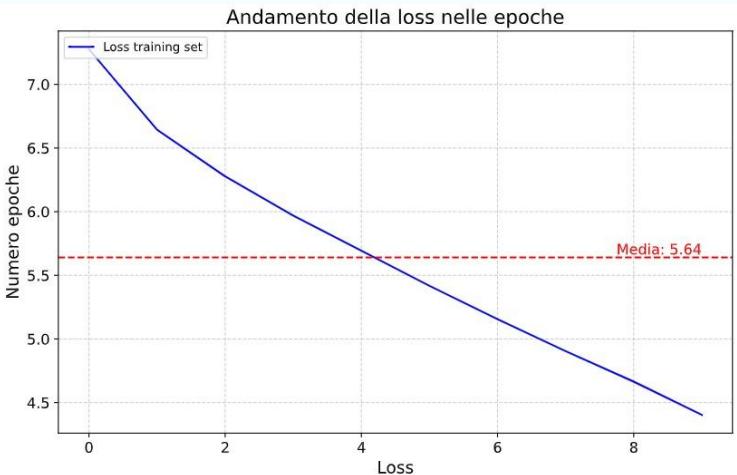


Risultati loss sulle epoche

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



Risultati loss grafico

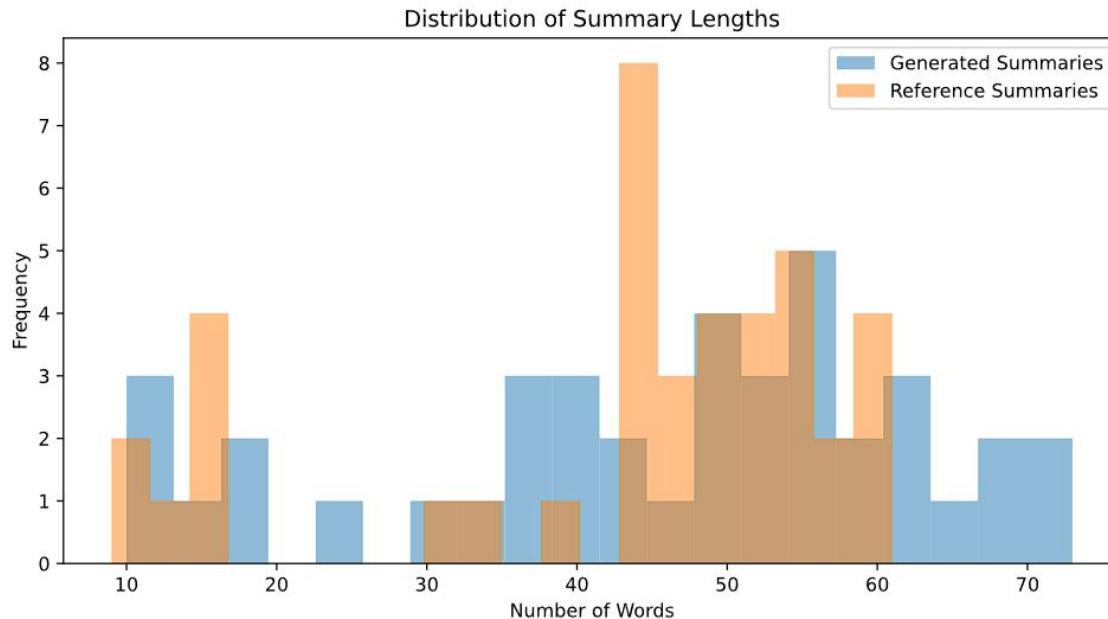


Osservazione

Un numero di epoche maggiori avrebbe comportato un miglioramento del modello.

L'implementazione dell'early stopping avrebbe garantito risultati ottimali, ma la potenza di calcolo era troppo limitata

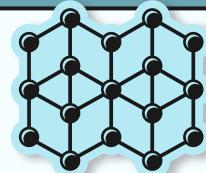
Distribuzione della lunghezza dei riassunti



05

Valutazione delle performance

Valutazione delle performance attraverso ROUGE e BLUE



ROUGE

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}
```

ROUGE-1, sovrapposizione di unigrammi, **ROUGE-2** sovrapposizione di bigrammi, **ROUGE-L** lunghezza della sequenza comune più lunga tra il testo generato e di riferimento.

BLUE

La misura BLUE è pensata per le traduzioni, varia tra 0 e 1, dove il valore 1 indica la perfetta corrispondenza tra testo generato e testo di riferimento, nel caso della sintesi non ha molto senso.

BLEU Score (intero corpus): 0.0604199308981629

Il BLUE Score rappresenta una misura complessiva della qualità della sintesi. Il valore 0.06 è molto basso, sia per la bassa accuratezza del modello, ma anche per l'errato utilizzo della metrica

Grazie per
l'attenzione

