UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

Fine-tuning di modelli di Abstractive Summarization su un dataset di articoli della conferenza ICDAR

Matteo Trippodo

Relazione laboratorio di Data Mining

Settembre, 2024

1 Introduzione

L'obiettivo principale di questa esperienza di laboratorio è stato quello di effettuare il fine-tuning di modelli di Deep Learning per un task di Abstractive Summarization su un dataset di articoli relativi alla conferenza Document Analysis and Recognition − ICDAR 2023. L'esperienza si è concentrata principalmente su due aspetti:

1. Utilizzare strumenti per recuperare i testi dei vari articoli a partire dai file pdf forniti dalla conferenza ;
2. Affinare l'addestramento di modelli per la creazione di riassunti su questo piccolo dataset, cercando di migliorare il più possibile le prestazioni dei modelli, sfruttando tecniche di ottimizzazione e regolarizzazione per evitare l'overfitting.

Il task di summarization appartiene al mondo della NLP e consiste nel creare un riassunto a partire da un articolo (o una porzione di esso), o in generale, da un testo fornito come input al modello. Il riassunto che è possibile generare grazie a modelli di deep learirng può essere di due tipi:

- Estrattivo (Extractive Summarization): in questo caso il modello apprende quali sono le frasi più importanti all'interno del contesto dell'articolo. Il riassunto generato è formato quindi dalle frasi più rappresentative che sono già presenti nell'articolo stesso.
- Astrattivo (Abstractive Summarization): i modelli di questo tipo tentano di creare un riassunto che non contenga frasi già presenti nell'articolo originario, andando quindi a modificare alcune parole o la struttura delle frasi più rappresentative, generando così un riassunto che dovrebbe essere più elaborato di quello del caso estrattivo.

Solitamente nel campo del Deep Learning applicato alla NLP (Natual Language Processing) vengono spesso sfruttate le tecniche del Transfer Learning e del Fine-tuning. Originariamente, lo studio e l'addestramento dei modelli prevedeva il partire da un modello da zero (*from scatch*), con pesi inizializzati in maniera casuale, secondo delle specifiche euristiche per fare in modo che i pesi iniziali semplificassero, o quando meno non facessero divergere, l'algoritmo di apprendimento (es. He/Xavier initialization).
Sebbene questo sia un approccio sempre possibile, con il passare del tempo, la grandezza dei modelli è cresciuta a dismisura, facendo aumentare di conseguenza le risorse e i dati necessari per l'addestramento di quest'ultimi. Per rendere tutto ciò più scalabile, evitando così di dover addestrare l'intero modello ogni volta su grandi quantità di dati, nasce l'idea del Trasfer Learning. Seguendo questo approccio, solo le grandi compagnie come Google, Microsoft, Facebook, si occupando di addestrare i modelli di deep learning su grandi dataset per task più generici, come quello dei language model.
I modelli cosidetti "pre-trained" vengono poi addestrati nuovamente su dataset più piccoli per downstream task più specifci. Questo secondo addestramento, il fine-tuning, è generalmente più veloce e più semplice, perché vengono usati meno dati e i pesi del modello sono già addestrati per performare correttamente in un task similare, pertanto non necessitano di un particolare aggiornamento e sono necessarie poche epoche per ottenere buone performance.

Generalmente, usando questo approccio si ottengono performance migliori che addestrando solamente un grande modello su un dataset più piccolo. Inoltre, l'uso di questa strategia è molto frequente nel mondo della NLP, poiché la prima fase di addestramento può essere spesso fatta in maniera non supervisionata (self-supervised), usando quindi semplicemente del testo come input, senza il bisogno di avere a disposizione un dataset etichettato.

2 Preprocessing e analisi del dataset

Il dataset di partenza è costituito da un gruppo di file in formato PDF che comprendono gli articoli presentati alla conferenza internazionale del 2023 nell'ambito della analisi e del riconoscimento di documenti, ICDAR (International Conference on Document Analysis and Recognition) per l'appunto.
A partire dai file in pdf è stato necessario utilizzare delle tecniche di preprocessing e cleaning per estrarre dai documenti della conferenza il testo dei vari singoli articoli, in modo che l'insieme dei documenti risultante sia il più "pulito" possibile, per aiutare l'apprendimento dei modelli considerati.
Per prima cosa è stato necessario, pertanto, estrarre il testo dai documenti; ciò può essere fatto in molteplici maniere o sfruttando varie librerie e tecniche. In questa esperienza di laboratorio è stata utilizzata la libreria Python PDFMiner che mette a disposizione il metodo *extract_ text()* che, passato come input il percorso del file, ritorna come output la stringa del testo estratto dal documento.
Tutto ciò è possibile in quanto un file pdf, e a maggior ragione un file contenente articoli scientifici, rappresenta una sorgente di dati **semi-strutturata**, in quanto di per sé sia un file pdf contiene una struttura data dal Page Description Language (PDL) usato per la sua formattazione e, a loro volta, i vari articoli hanno una loro struttura ben precisa, dettata dal fatto che anche essi devono sottostare a regole specifiche legate alla conferenza e ai suoi organi revisori, che analizzano gli articoli prima della pubblicazione e si assicurano che, oltre ai contenuti, anche la struttura degli articoli sia correttamente rispettata.
Sebbene non sia possibile sapere esattamente come l'estrazione del testo sia realizzata all'interno della libreria PDFMiner, generalmente vengono sfruttate tecniche di analisi del layout per la segmentazione del documento, in modo da individuare le zone del file contenenti il testo, assieme a tecniche di OCR (Optical Character Recognition) per individuare i singoli caratteri e comporre la stringa di testo in output.
Una volta usata la libreria otteniamo quindi una stringa di testo unica per tutto il file. A partire da essa è necessario estrarre il contenuto dei singoli articoli, e più nello specifico, per il task dell'abstractive summarization, siamo interessati a recuperare l'abstract, il corpo e, come spiegheremo poi, le conclusioni di ciascuno degli articoli. Il tutto è ottenuto sfruttando le potenzialità delle regex (espressioni regolari) che permettono di specificare, attraverso una opportuna sintassi, un pattern da ricercare all'interno di una stringa.
Sapendo, come annunciato in precedenza, che un articolo segue una specifica formattazione è possibile inferire l'inizio e la fine dell'abstract, così come l'inizio e la fine del corpo dell'articolo o delle conclusioni. I vari tre casi vengono gestiti in maniera simile ma con oppurtune accortezze a seconda del caso specifico:

- Abstract: l'abstract di qualsiasi articolo viene preceduto dalla stringa "Abstract." (l'inclusione del punto è importante perchè evita le possibili occorrenze della parola abstract all'interno dell'articolo stesso) e, generalmente, una volta concluso vengono elencate le parole chiave che rappresentano maggiormente gli argomenti contenuti nell'articolo. Pertanto, per trovare l'abstract è necessario definire un pattern che inizia con "Abstract." e finisce con la stringa "Keywords".
  Sebbene questo sia vero nella maggior parte dei casi, non tutti gli articoli dispongono di una sezione di parole chiave e, pertanto, è necessario gestire anche tutte le eventuali eccezioni. A tal fine la regex, in realtà, considera come finale opzionale per il pattern anche le stringhe "1 Introduction" o semplicemente "Introduction".
- Corpo: sfruttando una idea simile possiamo immaginare che ogni articolo inizi con una introduzione e presenti delle conclusioni finali. Pertanto, il modo più semplice per estrarre il corpo del testo è quello di utilizzare una regex per trovare nel testo un pattern che inizia con "Introduction" e finisce con "Conclusion". Ovviamente anche in questo caso, tutto non è così banale come potremmo immaginare, anche per un piccolo insieme di dati come quello considerato. Infatti, non tutti gli articoli hanno una vera sezione di conclusioni o più semplicemente utilizzano un titolo diverso per la sezione finale. Pertanto, come pattern alternativi per individuare la fine del corpo dell'articolo e quindi l'inizio delle conclusioni è necessario considerare anche "Discussion and Conclusion","Summary","Current Challenges","Result", o addiritura "Evaluation".
- Conclusioni: per quanto riguarda le conclusioni viene sfruttata nuovamente la conoscenza della struttura intrinseca del documento pdf e il fatto che sappiamo contenga i vari articoli uno di seguito all'altro. Pertanto, una volta individuati gli abstract dei vari documenti, che ne identificano il corrispettivo inizio, possiamo ridurre la porzione di testo su cui ricercare le conclusioni a quello contenuto tra la fine del corpo di un articolo e l'inizio del successivo. Da tale sotto-porzione di testo vengono estratti poi le conclusioni degli articoli andando alla ricerca del pattern "References" dopo la fine del corpo di un articolo.

Sfruttando queste logiche vengono estratti abstract, corpo e conclusioni dei testi contenuti nei vari file pdf, per un totale di 211 articoli ricavati. Abbiamo, pertanto, un dataset molto piccolo di dati, su cui siamo interessati a fare il fine-tuning di modelli di deep learning per la creazione di riassunti.
Comparate alle dimensioni dei vari modelli e quindi dal loro numero di parametri, seguendo la logica, ma anche più formalmente, considerando la teoria sottostante l'apprendimento e la modellazione di reti neurali, è facilmente comprensibile che un dataset di 211 documenti è molto esiguo per aspettarsi un modello altamente performante e che generalizzi correttamente. Infatti, come vedremo in seguito saranno utilizzate tecniche di regolarizzazione per evitare che i modelli overfittino tali dati.
É però utile anche specificare che una tecnica potenzialmente utile in questi casi è quella della Data Augmentation , ovvero aumentare artificiosamente la dimensione del dataset, andando a replicare più volte un campione, ma andandone a permutare alcuni aspetti, per esempio, alterando la struttura delle frasi mantenendone il significato o sostituendo una parola con un suo sinonimo. In questo modo, è come se avessimo dei campioni nuovi ma che sono ugualmente attendibili perché generati da quelli origina-

li applicando una qualche sorta di "rumore". In questa esperienza è stato fatto ricorso a tale tecnica, più nello specifico è stato tentato l'uso di sinonimi e la permutazione dell'ordine delle frasi in input, oltre che l'inserimento e la cancellazione casuale di parole. Rimane comunque uno strumento da tenere in considerazione anche per eventuali sviluppi futuri, poiché tecniche più raffinate potrebbero aiutare maggiormente l'addestramento dei modelli considerati.

Una volta estratte le varie sezioni degli articoli è stato importante verificare sperimentalmente la bontà dei testi estratti, infatti, molteplici volte i testi non erano partizionati correttamente, a causa della struttura del testo estratto da PDFMiner o dall'espressione regolare scritta impropriamente.
Tralasciando però eventuali errori, è inevitabile che in prima analisi il testo estratto dagli articoli contenga degli elementi indesiderati, che non aiutano o anzi mettono in difficoltà il potenziale addestramento di un modello. Alcuni esempi possono essere:

- i numeri di pagina
- i titoli di pagina, che spesso riprendono il titolo dell'articolo a cui fanno riferimento, assieme ad una citazione del nome del relativo autore principale
- le immagini e le relative didascalie
- le citazioni agli articoli nei riferimenti, esse possono seguire molteplici pattern ed è stato necessario valutare più volte i testi estratti per scoprirle tutte
- i dati contenuti in tabelle, che vengono comunque estratti ma devono essere ignorati
- collegamenti ipertestuali a siti o altri articoli

Nello specifico all'interno di ogni articolo viene inserita una dicitura del tipo:

Anche in questa occasione è stato necessario ricorrere all'uso delle espressioni regolari per identificare i pattern precedentemente mostrati e eliminarne il più possibile dal testo.
Conclusa questa prima fase di pulizia, per aiutare l'apprendimento del modello, il testo viene uniformato in modo da essere tutto lower case e evitando caratteri speciali, caratteri di a capo in eccesso e righe di testo vuote.
Una volta fatto tutto ciò è possibile creare il vero e proprio dataset, che sarà poi utilizzato per l'addestramento e la valutazione dei modelli, andando a creare prima un dataframe e poi i vari dataloader contenenti i token.

A termine di questa ampia descrizione del processo di preprocessing e cleaning effettuato sui testi estratti, è bene evidenziare che è stato seguito un approccio, per così dire, **best-effort**. Con questo si intende che è stato fatto tutto il possibile per togliere tut-

te quelle imperfezioni regolari contenute negli articoli, ma che, per motivi pratici e di bilanciamento del rapporto tra tempo necessario e qualità del risultati, questi non possono essere perfetti o all'altezza dei dataset ufficiali che è possibile trovare in circolazione. Ovviamente questo poiché l'ottenimento dei testi degli articoli non era il focus principale dell'esperienza di laboratorio, e pertanto, ci si è accontentati di una pulizia efficace, ma non totalmente completa dei testi, che però, inevitabilmente, può andare ad intaccare, anche solo parzialmente, la bontà dell'addestramento dei modelli considerati.

3 Modelli

Come detto in precedenza, in questa esperienza sono stati usati modelli di abstractive summarization, nello specifico vedremo l'uso di Bart, PropherNet e T5. Essi rappresentano tre architetture di modelli sequence-to-sequence che utilizzano blocchi encoder-decoder basati su transformer.
Per il task considerato i nostri modelli dovranno avere infatti come input l'articolo sorgente, o quanto meno parte di esso, e generare in output un testo che rappresenta il riassunto del testo fornito. Pertanto abbiamo bisogno che il modello trasformi la sequenza di parole in input in una sequenza di parole in output (modelli sequence-to-sequence appunto). Nei modelli considerati tale compito è svolto grazie alla presenza dei due componenti principali della rete: l'encoder e il decoder.
L'encoder prende come input gli embedding (le rappresentazioni numeriche) delle parole (token) contenute nell'articolo ed è in grado di elaborarne il contenuto in modo da generare un "contesto", ossia una rappresentazione astratta (latente) del contenuto semantico del testo passato in input. Il decoder prende questo contesto generato dall'encoder e una volta passatogli un token di inizio sequenza questo genera il riassunto una parola alla volta. Questo è possibile grazie all'ultimo layer del modello che realizza una vera e propria classificazione sullo spazio degli embedding (e quindi dei token) per determinare quale è la prossima parola più probabile dato il contesto dell'encoder e le parole generate precedentemente. Encoder e decoder sono costituiti da molteplici blocchi di transfomer che a seconda dell'architettura avranno una struttura e una dimensione leggermente diversa.
In generale, un blocco di transformer, come accade ormai in tutti i campi del Deep Learing, per favorire la modularità e allo stesso tempo la flessibilità, può essere realizzato in vari modi, ma è sempre costituito concatenando alcuni elementi fondamentali:

- Self-attention layers: permettono di valutare le relazioni tra i token presenti all'interno di una stessa sequenza. Tramite una elaborazione legata a pesi e combinazioni dei valori degli embedding stessi, questi layer permettono di determinare l'importanza di una parola all'interno di una frase e quanto le restanti parole siano collegate ad essa. Ciò è possibile perché i modelli rappresentano le parole come degli embedding, ovvero vettori le cui feature vengono apprese in modo tale da racchiuderne il contenuto semantico, di modo che parole simili, o che appartengono allo stesso contesto, abbiano embedding vicini, secondo una qualche forma di similarità.
- Batch Normalization layers: permettono di stabilizzare l'apprendimento del modello, andando a normalizzare i valori passati in input riportandoli in un range

che facilita l'addestramento del modello e evita la divergenza dell'algoritmo di ottimizzazione utilizzato.

- Feed-Forward layers: sono costituiti da più strati di fully-connected layers che introducono non linearità e permettono di apprendere complesse relazioni tra i token in una sequenza.
- Connessioni residuali: consistono nel andare a sommare all'output direttamente l'input di un blocco. Generalmente, si ha una connessione residuale che collega ingresso e uscita di un self-attention layer o di un feed-forward layer. In questo modo il modello deve solo apprendere la differenza tra l'input e l'output desiderato, andando pertanto a semplificare molto l'addestramento e a stabilizzarlo, evitando inoltre il vanishing del gradiente.

É utile anche analizzare nel dettaglio i vari modelli considerati, in modo da comprenderne le similarità e le differenze, che si ripercuoteranno poi, inevitabilmente, in particolari accortezze necessarie durante l'addestramento e differenti performance in termini di riassunti generati. Vediamo quindi le caratteristiche principali dei vari modelli:

- BART (Bidirectional and Auto-Regressive Transformers): è un modello sequence-to-sequence addestrato da Facebook AI che realizza un cosiddetto "denoising autoencoder". Questo significa che è stato addestrato fornendo in input una versione corrotta (rumorosa) del testo e il modello deve essere in grado di ricostruire la versione originale. Usualmente, quello che viene fatto è che alcuni dei token vengono mascherati e il modello deve imparare a predirli correttamente. Inoltre, come possiamo capire dal nome, il modello all'interno dell'encoder sfrutta una tipologia di attenzione che è bidirezioanale, ovvero determina l'importanza di un token all'interno di una sequenza andando a elaborare i token in entrambe le direzioni. Pertanto, l'importanza di un token all'interno di una frase dipende sia dai token precedenti che quelli successivi. In più, come si intuisce, BART all'interno del decoder utilizza una tecnica auto-regressiva per la generazione del riassunto, ovvero una volta che il decoder genera il token successivo più probabile questo viene rifornito in input per predire il seguente.
  Il modello base di Bart, ossia quello utilizzato in questo elaborato, ha circa 140 milioni di parametri.
- T5 (Text-To-Text Transfer Transformer): è modello seq2seq addestrato da Google su molteplici task, infatti, la stessa rete può essere impiegata sia per generare riassunti che per la traduzione di testi. Infatti, per ottenere i migliori risultati il modello ha bisogno del prompt "summarize:" per comprendere che l'obbiettivo dell'addestramento è generare un riassunto. Questo perché T5 è stato addestrato su molteplici task che posso essere considerati in un formato "text-to-text", cercando di risolvere così più problemi utilizzando un approccio condiviso, che sfrutta poi un prompting per triggerare il comportamento desiderato. T5 nasce per sfruttare il principio del trasfer learning, ovvero prima il modello viene addestrato per un task generico (in questo caso come language model) su un ampio dataset di dati e poi l'addestramento viene affinato (fine-tuning) su downstream task secondario (ad esempio la generazione di riassunti).
  La versione di T5 considerata, quella base, ha circa 222 milioni di parametri.

- ProphetNet: è un modello seq2seq di Microsoft che sfrutta il transfer learning, utilizzando un pre-addestramento non supervisionato come language model e poi un fine-tuning su due downstream task, abstractive summarization e generazione di domande. La caratteristica principale che differenzia ProphetNet dai modelli precedenti è che sfrutta un addestramento basato sulla predizione dei futuri n-grammi (e non del singolo token) e un meccanismo di attenzione basato su n-stream. Infatti, ProphetNet è ottimizzato per fare predizioni di n step nel futuro, generando i successivi n token simultaneamente sulla base del contesto dato dal gruppo di token precedenti. La rete contiene uno stream principale di self-attention (quello che sarebbe il flusso usato normalmente negli altri modelli basati su transformers) e introduce in più ulteriori stream di self-attention per la previsione dei futuri rispettivi n-grammi. Questo è ciò che accade nella fase di training, però, l'implementazione della rete è stata fatta in modo tale che durante l'inferenza i token vengano generati uno alla volta. ProphetNet è il modello più grande tra i tre, con quasi 400 milioni di parametri e il suo pre-training è effettuato su un grande dataset di circa 160 GB di dati.

4 Fine-tuning e risultati

Il fine-tuning dei modelli indicati è stato eseguito considerando due diversi setting: per prima cosa utilizzando unicamente i primi 512 token dell'articolo per l'addestramento e, in seconda battuta, utilizzando i primi 256 token dell'articolo suguiti dai primi 256 token delle conclusioni.
In più, visto che le dimensioni del dataset sono molto limitate, vengono valutate le prestazioni andando a considerare l'aggiunta di tecniche di regolarizzazione come dropout e weight decay durante la fase di addestramento, oltre all'utilizzo di strategie di data augmentation.
Le performance dei vari modelli sono state valutate sulle principali metriche utilizzate nel campo della NLP, e nello specifico per il caso di generazione dei riassunti, ovvero le metriche rouge e le metriche blue, usate per valutare la bontà dei riassunti e la relativa n-gram novelty.

Come visto in precedenza, per l'addestramento dei nostri modelli abbiamo a disposizione un dataset costituito da 211 articoli, che risulta essere molto piccolo se rapportato alle dimensioni dei modelli considerati, che sono costituiti da centinaia di milioni di parametri. Pertanto, possiamo aspettarci che difficilmente riusciremo ad ottenere valori troppo bassi di loss di training e validazione o, in generale, grandi prestazioni sul testing set. Infatti, è bene sottolineare che, al fine di un addestramento corretto del modello, è necessario suddividere il dataset in 3 split: training set, validation set e test set.
Al fine di avere un sufficiente numero di articoli per l'addestramento, ma allo stesso tempo avere anche abbastanza articoli per poi valutare le prestazioni di test, viene considerato un train/test split del 90%. In questo modo 21 articoli sono lasciati per valutare le performance finali, mentre i restanti 190 vengono utilizzati durante l'addestramento. Di questi, solo 169 sono realmente utilizzati per l'addestramento effettivo (ovvero utilizzati per calcolare la loss e aggiornare i parametri del modello), mentre i

restanti 21 sono utilizzati per il set di validazione (in questo modo tale insieme è della stessa dimensione di quello di test, in modo da poter direzionare sperabilmente meglio l'addestramento verso risultati migliori).
Il dataset di validazione è fondamentale per l'addestramento di modelli di deep learning perché ci permette di valutare le prestazioni di generalizzazione del nostro modello durante la fase di addestramento. Infatti, se concentriamo l'apprendimento sul mero obiettivo di minimizzare la loss di training finiremo facilmente per overfittare i dati di input. Invece, monitorare le performance su un dataset esterno, quello di validazione, ci fornisce degli indizi per modificare gli iperparametri scelti per l'addestramento o terminarlo prima che esso cada nell'overfitting (early stopping).

Il fine-tuning di un modello, ma in generale anche l'addestramento, non segue delle regole ben precise, ma più che altro delle euristiche e delle buone pratiche che sono state scoperte negli anni dai ricercatori nel campo del Deep Learning. Pertanto, come è accaduto in questa esperienza di laboratorio, prima di trovare una procedura di addestramento che può essere considerata soddisfacente, molte volte sono stati necessari degli approcci "trial and error" in cui si provano varie tecniche e si valuta poi quale risulta essere la più efficace. In questa esperienza di laboratorio, tale approccio è stato utilizzato per valutare empiricamente quale approccio di addestramento utilizzare per il fine-tuning dei modelli considerati. Infatti, quando si addestra un modello di deep learing è necessario scegliere il valore ottimale di una serie di iperparametri (chiamati in questo modo perché influenzano in maniera indiretta le performance del modello in quanto ne determinano l'apprendimento). Nel caso considerato è necessario scegliere:

- Ottimizzatore: rappresenta l'algoritmo di ottimizzazione da utilizzare per aggiornare i parametri del modello durante l'addestramento. Le principali scelte, o comunque quelle più usualmente utilizzate, sono SGD (Stochastic Gradient Descent) e Adam (Adaptive Moment Estimation). In questo caso è stato preferito Adam come ottimizzatore poiché permette di aggiornare i pesi del modello tenendo conto di un termine di momemntum, che regola la velocità con cui un batch di dati direziona l'algoritmo di ottimizzazione verso il desiderato minimo globale. Inoltre, Adam a differenza di SGD mantiene un differente learning rate per ciascuno dei parametri che viene aggiornato considerando i momenti del primo e del secondo ordine del gradiente. É stato testato anche un addestramento con SDG e momentum di Nestorov, ma Adam sperimentalmente sembra garantire una stabilita migliore per l'addestramento e per i risultati finali di loss di validazione.

- Learning Rate: rappresenta il fattore moltiplicativo applicato al valore del gradiente che viene usato durante l'aggiornamento dei pesi del modello. Utilizzare un learning rate troppo alto porta l'algoritmo di ottimizzazione a divergere, un valore troppo piccolo invece porta a fare aggiornamenti troppo piccoli ai pesi durante l'addestramento ottenendo ugualmente ad modello poco addestrato o ad una soluzione sub-ottimale. Ovviamente, non esiste un valore ottimale da utilizzare in quanto quello che è il valore più efficace dipende la task specifico e dal modello che consideriamo. Per modelli di grandi dimensioni come quelli considerati è necessario utilizzare dei learning rate molto piccolo nell'ordine di 10^{-5}. Visto che i modelli hanno dimensioni diverse è opportuno variare leggermente il learning rate per adattare correttamente l'addestramento al modello considerato. In più, so-

litamente, per favorire l'addestramento può essere utile definire un learning rate schedule, che descrive come il valore del lr evolve durante il passare delle epoche. Esistono tante tipologie di schedule per il lr, come lineare, polinomiale, cosinusoidale, basata su step o anche con warm up (nelle epoche iniziali il learning rate viene aumentato in modo tale da velocizzare la convergenza dell'addestramento). In questa esperienza di laboratorio è stato sfruttato principalmente una step lr scheduler che permette di ridurre il learning rate di un certo fattore dopo un definito numero di epoche. Pertanto, questa tipo di schedule ha 2 iper-parametri da poter ottimizzare (step size e gamma).

Al fine di velocizzare parzialmente questo processo è stata utilizzata talvolta la libreria Optuna che consente di specificare gli iperparametri di interesse da ottimizzare e degli intervalli in cui tentare i valori. Una volta fissato i numero di tentativi, per ognuno di essi la libreria considera una combinazione di iperparametri e addestra il modello, ritornando alla fine il set di parametri che ha minimizzato una certa metrica di interesse, nel nostro caso la validation loss. Essendo che però fare il fine-tuning dei modelli considerati richiede molti minuti, non è stato possibile utilizzare intensivamente Optuna e quindi sfruttarlo al meglio, in quanto per fare uno studio con 20 tentativi erano necessarie più di 3 ore.

Come vedremo, non è stato possibile ottenere prestazioni troppo elevate per nessuno dei modelli, sia in termine di metriche rogue che n-gram novelty, anche dopo numerosi tentativi. Questo deve essere principalmente dovuto al fatto che i dati utilizzati sono molto limitati e, nonostante i modelli siano già pre-addestrati su grandi quantità di dati, non consentono al modello di generare riassunti molto accurati. Questo fa riflettere sulla complessità del task della summarization e di come grandi modelli abbiamo bisogno di enormi quantità di dati per performare correttamente.

Le metriche valutate sono principalmente le metriche rogue e le metriche blue:

- Le metriche ROUGE (Recall-Oriented Understudy for Gisting Evaluation) sono una serie di metriche di valutazione automatica ampiamente utilizzate nel campo del NLP. Queste metriche posso essere utilizzate, quindi, per valutare la qualità dei riassunti generati confrontandoli con i riassunti di riferimento definiti dagli abstract dei vari articoli.
  Le metriche ROUGE sono principalmente basate su confronti di n-grammi e sequenze di parole tra il riassunto di riferimento e il riassunto generato.
  Le principali metriche sono ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S, ROUGE-SU, ecc. Le più comuni e utilizzate sono però ROUGE-1 (unigram), ROUGE-2 (bigram) e ROUGE-L, e sono pertanto quelle su cui si è concentrata l'esperienza di laboratorio.
  Nel nostro caso viene considerato come metrica di riferimento il F1 score, ovvero la media armonica tra precisione e recall, mediato tra tutti gli n-grammi.
- Le metriche BLEU (Bilingual Evaluation Understudy) nascono per valutare la qualità delle traduzioni automatiche, ovvero di modelli di Machine Traslation (come può essere ad esempio lo stesso T5), per poi essere estese e impiegate per la valutazione di tutti quei modelli che hanno task relativi al campo NLP.

I blue score assumono valori tra 0 e 1 e rappresentano il livello di corrispondenza tra due testi considerati, dove uno sarà quello di riferimento e l'altro quello generato dal modello che vogliamo valutare.
Per le varie dimensioni n degli n-grammi, le metriche blue contano il numero di volte che un certo n-gramma appare nel testo di riferimento e, quindi, per ognuno di esse, si calcola la n-gram precision, come il rapporto tra quante volte esso si presenta nel testo generato e quante volte in quello di riferimento. Il valore finale della metrica viene poi calcolato facendo una media geometrica in cui la precision di ogni n-gramma è moltiplicata per un per un fattore di Brevity Penalty, inversamente proporzionale alla lunghezza della frase.
Nel nostro caso, le metriche blue possono essere sfruttate per calcolare la n-gram novelty tra due frasi. Infatti, di base le metriche blue possono essere utilizzare per valutare la sovrapposizione tra due frasi. Conseguentemente, il complementare ad 1 di tali score può essere interpretato come la percentuale di "novità" di una frase, nel nostro caso quella generata dal modello, rispetto ad un'altra, quella del riassunto target.
Nel nostro caso, viene prima calcolato lo score per parole, bigrammi e trigrammi, e considerato poi il complementare per avere il valore di n-gram novelty.
Inoltre, per il calcolo del blue score è stata usata una smoothing function basata sul cosiddetto method 1, che consiste nell'assegnare un certo epsilon (generalmente 0.1) alla precision di quegli i n-grammi che altrimenti avrebbero zero. Ciò risulta necessario, soprattutto per n-grammi di grandi dimensioni, poiché a volte abbiamo una bassissima sovrapposizione e in tal modo evitiamo di avere un blue score finale nullo.

Approfondiamo l'analisi dei vari modelli:

1. BART

Considerando le risorse disponibili su Google Colab, per addestrare BART per un'epoca sono necessari circa 120 secondi. Pertanto, il tempo di addestramento non è estramamente elevato e ha consentito di fare numerosi test con diversi setting e iperparametri. Un buon addestramento è possibile introducendo una prima fase di warmup per il learing rate e poi riducendono usando una step decrease schedule, ad esempio partendo con un learning rate di 10^{-5} e un gamma di 0.6.

Come vediamo dai risultati numerici, nel caso di BART, le performance in termini di rouge score sono sempre intorno allo 0.4 per il ROGUE-1, intorno allo 0.1 per il ROGUE-2 e 0.2 per il rouge-L.
I valori di rouge-1 sono abbastanza adeguati e significano che c'è una discreta sovrapposizione tra le parole usate all'interno dell'abstract e del riassunto generato, però come dimostrano le altre due metriche i riassunti non riescono a cogliere relazioni più ampie come tra coppie di parole o frasi. Quindi, i riassunti generati riescono a cogliere parole chiave ma non a generare riassunti simili a quelli di target. Ciò è ancora più evidente se guardiamo i valori di n-gram novelty, infatti, come si vede questi sono tutti abbastanza alti. Infatti, se per le singole parole (unigram) abbiamo valori minori, perché comunque parte delle parole chiave sono integrate nei riassunti generati, per quanto riguarda bi-

gram e trigram otteniamo una n-gram novelty alta, sintomo di sovrapposizione molto bassa tra gli n-gram generati e quelli di riferimento.
Se tendenzialmente valori abbastanza alti di novelty rappresentino il fatto che il modello rielabora molto il contenuto del testo e genera un riassunto alternativo, valori troppo alti possono essere dovuti al fatto che il riassunto generato non sia sufficientemente congruente con quello di riferimento.

In aggiunta, analizzando i dati delle tabelle vediamo come, in generale, l'uso dei parte delle conclusioni permette di migliorare leggermente le prestazioni per tutte le metriche rouge. In più, l'uso di tecniche di data augmentation consente di migliorare lievemente le prestazioni, anche senza l'uso di conclusioni, mentre, invece, l'uso di dropout sembra non essere efficacie per aiutare il modello a generalizzare meglio, ma anzi porta a vanificare gli effetti della data augmentation, rendendo inoltre sperimentalmente l'addestramento più complesso. Per tale ragione nel caso dei prossimi modelli, verrà considerato unicamente l'uso della data augmentation, poiché l'introduzione di dropout, anche leggero, sembra destabilizzare l'apprendimento dei modelli, che nel caso di T5 e Prophetnet è un processo ancora più lento e complesso.

Come vediamo dagli esempi dei riassunti generati, BART genera un riassunto piuttosto sensato e che segue coerentemente l'articolo , senza però seguire esattamente il contenuto del riassunto target (abstract). Quello che si può notare infatti è che le prime frasi del riassunto riprendono spesso quasi esattamente l'inizio del testo fornito come input. Se in parte questo comportamento può essere dovuto al fatto che il modello è addestrato su un dataset limitando, in parte però tale fenomeno è sicuramente influenzato anche da come BART è stato pre-addestrato. Pertanto, sperimentalmente possiamo notare come talvolta BART non rielabora molto le frasi e genera un riassunto che è quasi più estrattivo che astrattivo.

2. T5

Utilizzando le risorse di Colab, per fare un epoca con T5 si impiega circa 200 secondi; ciò riflette le maggiori dimensioni di questo modello rispetto al BART. Per l'addestramento di T5 è stato utilizzato un approccio simile al precente; sperimentalmete è risultato più utile partire con un learning rate leggermente più elevato, ad esempio $5x10^{-5}$.

Come vediamo nel caso di T5 ottieniamo prestazioni simili al caso precedente , la metrica rouge-1 sempre vicino a 0.4, mentre le restanti metriche rouge abbastanza basse, e allo stesso tempo le metrice per la n-gram novelty assumono valori alti.
In generale, i valori sembrano essere leggermente peggiori rispetto alle performance viste per BART, nonostante T5 sia un modello più grande e che ha rappresentato lo stato dell'arte per un buon periodo per task NLP. Questo potrebbe essere dovuto al fatto che con i pochi dati a disposizione T5 non riesce sfruttare al meglio le proprie potenzialità, finendo per performare peggio in termini numerici.
Vediamo comunque però che l'uso di tecniche di data augmentation consentono di aumentare le performance per tutte le metriche considerate, così come anche l'uso delle conclusioni risulta efficace in questo caso, soprattuto del caso senza data augmentation.

Nonostante la leggera differenza di performance, anche in questo caso il modello sembra generare un riassunto coerente con il target desiderato. Possiamo vedere come alcune frasi inserite nel riassunto generato sono delle leggere rielaborazione di frasi contenute nel testo fornito in input al modello. Anche in questo caso, la parte iniziale dei riassunto riprende il contenuto iniziale dell'articolo, ma in maniera minore rispetto a T5. Sperimentalmente, i riassunti generati da T5 sembrano essere leggermente migliori e più astrattivi rispetto a quelli di BART.

2. Prophetnet

Questo modello, essendo molto grande per completare un epoca di addestramento, considerando le risorse disponibili su Google Colab, sono necessari più di 600 secondi. Inoltre, anche l'inferenza, ossia la generazione del riassunto risulta essere molto più lenta rispetto ai modelli precedenti. Ciò si ripercuote su un elevato tempo necessario anche per testare le performance del modello e valutare le metriche rouge e blue. Pertanto, non è stato possibile sperimentare troppi valori diversi per gli iperparametri, anche se generalmente l'addestramento perdurava un numero minore di epoche rispetto ai casi precedenti.

Come si vede dai risultati, le metriche assumono valori similari a quelli dei casi precedenti, però in generale Prophetnet sembra essere il modello che fornisce le performance migliori tra i tre considerati. Infatti, sebbene il valori delle metriche rouge siano solo leggermente maggiori dei casi precedenti, è importante notare come i valori di n-gram novelty, sopratutto per il caso degli unigram, risultano essere considerevolmente inferiori rispetto a quanto visto per gli altri modelli, prova del fatto che Prophetnet segue maggiormente il target desiderato, generando così dei riassunti più congruenti.
Anche in questo caso l'utilizzo delle conclusioni consente di ottenere valori migliori e principalmente per le metriche rouge. Inoltre, anche l'uso di data augmentation sembra dare i suoi frutti portando un miglioramento per tutte le metriche considerate.

Guardando agli esempi di riassunti generati, abbiamo una prova del fatto che Prophetnet a differenza dei due modelli precedenti risulta essere maggiormente in grado di generare riassunti accurati. Infatti, i riassunti generati riprendendono spesso alcune delle parole chiave contenute negli abstract senza però copiare il contenuto del testo di input, come in parte facevano i modelli precedenti.

5 Conclusioni

In conclusione, abbiamo visto come è possibile fare il fine-tunining di modelli pre-addestrati per un task di abstractive summarization e come verificato sperimentalmente come esso non sia un compito affatto facile. Nel caso in questione, il dataset utilizzato per affinare i modelli, essendo relativo ad unica conferenza era molto limitato. Questo ha reso molto complesso ottenere buone prestazioni dai modelli che hanno raggiunti tutti discreti valori R-1 score, ma con valori abbastanza bassi per le restanti metriche.
É stato testando in parte l'uso di dropout, che però è risultato essere inefficace, mentre tramite lo sfruttamento di tecniche di data augmentation è stato possibile migliorare le performance nella maggior parte dei casi. Inoltre, anche l'inclusione di parte delle

conclusioni durante la fase di addestramento sembra aiutare ai modelli ad apprendere meglio quali sono le informazioni rilevanti, consentendo di generare riassunti migliori e più accurati. Infatti, spesso le informazioni più importanti, che vengono solitamente espresse nell'abstract, non spesso riportate nella parte iniziale dell'articolo che funge da introduzione, ma bensì esse sono spesso presenti nelle conclusioni. Pertanto, includere le conclusioni aiuta a generare riassunti più simili agli abstract di riferimento.

Pertanto, quanto ne consegue, è che per addestrare correttamente modelli di grandi dimensioni come quelli trattati sono necessari un maggior numero di dati, sicuramente anche di qualità migliore, in ordine di consentire al modello di avere una maggiore variabilità dei dati e quindi di generalizzare meglio, realizzando riassunti più coerenti anche per nuovi articoli. Ciò nonostante, come dimostrato in precedenza, l'utilizzo di tecniche come quelle considerate permette di aumentare le prestazioni, anche in scarsità di dati.

6 Appendice

6.1 Performance

Model	w./o concl.			w. concl.		
	ROGUE-1	ROGUE-2	ROGUE-L	ROGUE-1	ROGUE-2	ROGUE-L
BART	0.44301	0.13184	0.21600	0.45100	0.15289	0.23845
BART w. Dropout	0.42705	0.11942	0.21143	0.44255	0.15031	0.22852
BART w. DA	0.44313	0.12743	0.21504	0.45588	0.146590	0.24322
BART w. Dropout + DA	0.42230	0.12124	0.21820	0.44665	0.14655	0.24816

Model	w./o concl.			w. concl.		
	1-gram novelty	2-gram novelty	3-gram novelty	1-gram novelty	2-gram novelty	3-gram novelty
BART	0.68687	0.91465	0.96178	0.71436	0.91392	0.96052
BART w. Dropout	0.693875	0.922819	0.973247	0.716856	0.9254395	0.972869
BART w. DA	0.67531	0.91387	0.96831	0.72007	0.92489	0.97348
BART w. Dropout + DA	0.71296	0.931113	0.974336	0.7256403	0.9219643	0.96330

Tabella 1: ROUGE score e N-gram novelty per fine-tuned BART

Model	w./o concl.			w. concl.		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
T5	0.40899	0.13192	0.22503	0.42549	0.13950	0.24297
T5 w. DA	0.43132	0.13746	0.23223	0.43428	0.13816	0.22873

Model	w./o concl.			w. concl.		
	1-gram novelty	2-gram novelty	3-gram novelty	1-gram novelty	2-gram novelty	3-gram novelty
T5	0.76217	0.92328	0.96189	0.75735	0.92734	0.96490
T5 w. DA	0.71212	0.91956	0.96235	0.75270	0.93040	0.96990

Tabella 2: ROUGE score e N-gram novelty per fine-tuned T5

Model	w./o concl.			w. concl.		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
Prophetnet	0.45219	0.12639	0.21647	0.47061	0.14707	0.21403
Prophetnet w. DA	0.47007	0.14440	0.22577	0.48379	0.15907	0.22585

Model	w./o concl.			w. concl.		
	1-gram novelty	2-gram novelty	3-gram novelty	1-gram novelty	2-gram novelty	3-gram novelty
Prophetnet	0.61415	0.89140	0.94801	0.61489	0.87763	0.93725
Prophetnet w. DA	0.61462	0.88678	0.94558	0.60814	0.87768	0.94322

Tabella 3: ROUGE score e N-gram novelty per fine-tuned ProphetNet

6.2 Esempi di riassunti generati

Di seguito sono mostrati alcuni esempi di come i vari modelli effettuano i loro riassunti. Sono stati considerati 3 articoli appartenenti all'insieme di test e per ognuno di essi viene indicato ciò che viene passato in input al modello, ovvero la parte iniziale del corpo dell'articolo, assieme all'abstract originale e successivamente cosa generano in output i tre differenti modelli. I modelli sono addestrati integrando le conclusioni nell'input e applicando data augmentation.
Del testo originale sono sottilineate le parti che vengono utilizzate dai modelli nei loro riassunti. Queste costituiscono le parte della sequenza che risultano essere più importanti tramite i meccanismi di attention all'interno dei modelli addestrati.

Articolo n.1

Testo originale:
the rey-osterrieth complex figure test (rocft) was designed to examine the visuospatial ability and memory in patients who suffer from traumatic brain injury. additionally, the test is utilized to test for dementia and to evaluate children's cognitive development. the test procedure starts with presenting the figure depicted in fig. 1 to the patient, who is subsequently asked to copy it by drawing it, typically with a pen on paper. after 3 min, the patient is asked to reproduce the figure from memory. this procedure is repeated after 30 min. b. schuster et al. those three steps are called copy, immediate recall, and delayed recall. while copy is always part of the procedure, sometimes only one of the steps immediate recall or delayed recall are carried out. the figure can be subdivided into 18 separate sections, as annotated in fig. 1. according to the osterrieth scoring system, for each section, a score ranging from 0 to 2 is determined in the following way: if the unit is drawn and placed correctly, 2 points are assigned. in case it is placed poorly, this corresponds to in this work, we developed and validated an approach for the multi-stage fine-tuning of deep learning models for an automatic assessment of the reyosterrieth complex figure test. specifically, we studied and combined (1) comprehensive data augmentation strategies, (2) pre-training on visually related sketch images, (3) swa for improved model generalization to unseen image characteristics, and (4) ensembling with different model and augmentation combinations. we could experimentally verify that using an application-driven fine-tuning strategy on visually similar but semantically contrasting sketch images can substantially improve the error rate. this improvement is not only shown in direct comparison with a fine-tuning strategy on imagenet, but also holds for almost all network architectures studied. this means that additional data collection and curation can be avoided with relatively little effort. intriguingly, applying swa shows a small but recognizable improvement. since this technique is applied in a separate, very much shortened fine-tuning step, it imposes a negligible additional computational burden. in stark contrast, ensembling multiple network models represents a larger cut in the system's efficiency

Abstract:
the rey-osterrieth complex figure test (rocft) is a widely used neuropsychological tool for assessing the presence and severity of different diseases. it involves pre-

senting a complex illustration to the patient who is asked to copy it, followed by recall from memory after 3 and 30 min. in clinical practice, a human rater evaluates each component of the reproduction, with the overall score indicating illness severity. however, this method is both time-consuming and error-prone. efforts have been made to automate the process, but current algorithms require large-scale private datasets of up to 20,000 illustrations. with limited data, training a deep learning model is challenging. this study addresses this challenge by developing a fine-tuning strategy with multiple stages. we show that pre-training on a large-scale sketch dataset with initialized weights from imagenet significantly reduces the mean absolute error (mae) compared to just training with initialized weights from imagenet, e.g., rexnet-200 from 3.1 to 2.2 mae. additionally, techniques such as stochastic weight averaging (swa) and ensembling of different architectures can further reduce the error to an mae of 1.97.

Riassunto generato da BART:
the rey-osterrieth complex figure test (rocft) was designed to examine the visuospatial ability and memory in patients who suffer from traumatic brain injury. additionally, the test is utilized to test for dementia and to evaluate children's cognitive development. in this work, we propose a multi-stage fine-tuning of deep learning models for an automatic assessment of the figure. we evaluate the performance of our approach on visually similar but semantically contrasting sketch images. we demonstrate that our approach achieves a small but recognizable improvement in the error rate on visually related sketch images compared to previous methods.

Riassunto generato da T5:
the rey-osterrieth complex figure test (rocft) is designed to examine the visuospatial ability and memory in patients who suffer from traumatic brain injury. the test procedure starts with presenting the figure to the patient, who is then asked to copy it by drawing it, typically with a pen on paper. after 3 min, the patient is asked to reproduce the figure from memory, which is repeated after 30 min. in this work, we developed and validated an approach for the multi-stage fine-tuning of deep learning models for an automatic assessment of the rocf.

Riassunto generato da ProphetNet:
the rey-osterrieth complex figure test (rocft) aims to assess the visuospatial ability and memory in patients who from traumatic brain injury. in this work, we investigate and validate a multi-stage fine-tuning strategy for deep learning models to generalize to visually similar but semantically contrasting sketch images. specifically, we study and together study and combine (1) comprehensive data augmentation strategies, (2) pre - training on visually related sketch images, and (3) ensembling multiple network models to improve model generalization to unseen image characteristics. we show that applying an application - driven fine tuning strategy on visually similar sketch images can substantially improve the error rate of the rocft.

Articolo n.2

Testo originale:
handwritten mathematical expression recognition (hmer) is an important ocr task, which can be widely applied in question parsing and answer sheet correction. in recent years, with the rapid development of deep learning technology, scene text recognition approaches have achieved great progress. however, due to the ambiguities brought by crabbed handwriting and the complicated structures of handwritten mathematical expressions, hmer is still a challenging task. fig. 1. illustration of our method. the different colored graph nodes and arrows indicate different symbols. built upon the recent progress in sequence-to-sequence learning and neural networks, some studies have addressed hmer with end-to-end trained encoder-decoder models and showed significant improvement in performance. nevertheless, the encoder-decoder framework do not fully explore the correlation between different symbols in the mathematical expression, which may be struggling when facing similar handwritten symbols or crabbed handwritings. to address above issues, we argue that an effective hmer model should be improved from the following two aspects: (1) capturing semantic dependencies among different symbols in the mathematical expression; (2) integrating more semantic information to locate the regions of interest. in this paper this paper has presented a simple and efficient method for handwritten mathematical expression recognition by incorporate semantic graph representation learning into end-to-end training. to our best knowledge, the proposed method is the first to learn the correlation between different symbols through symbol cooccurrence probabilities. experiments on the crohme dataset and hme100k dataset have validated the effectiveness and efficiency of our method. acknowledgement. this work was supported by national key r&d program of china, under grant no. 2020aaa0104500 and national science fund for distinguished young scholars of china (grant no.62225603).

Abstract:
handwritten mathematical expression recognition (hmer) has attracted extensive attention recently. however, current methods cannot explicitly study the interactions between different symbols, which may fail when faced similar symbols. to alleviate this issue, we propose a simple but efficient method to enhance semantic interaction learning (sil). specifically, we firstly construct a semantic graph based on the statistical symbol co-occurrence probabilities. then we design a semantic aware module (sam), which projects the visual and classification feature into semantic space. the cosine distance between different projected vectors indicates the correlation between symbols. and jointly optimizing hmer and sil can explicitly enhances the model's understanding of symbol relationships. in addition, sam can be easily plugged into existing attention-based models for hmer and consistently bring improvement. extensive experiments on public benchmark datasets demonstrate that our proposed module can effectively enhance the recognition performance. our method achieves better recognition performance than prior arts on both crohme and hme100k datasets.

Riassunto generato da BART:

written mathematical expression recognition (hmer) is an important ocr task, which can be widely applied in question parsing and answer sheet correction. in recent years, with the rapid development of deep learning technology, scene text recognition approaches have achieved great progress. however, due to the ambiguities brought by crabbed handwriting and the complicated structures of handwritten mathematical expressions, hmer is still a challenging task. to address these issues, we propose a simple and efficient method to learn the correlation between different symbols in the mathematical expression by integrating semantic graph representation learning into end-to-end training. to our best knowledge, the proposed method is the first to learn symbol cooccurrence probabilities through symbol cooperation probabilities. experiments on the crohme dataset and hme100k dataset show significant improvement in hmer performance.

Riassunto generato da T5:
handwritten mathematical expression recognition (hmer) is an important ocr task, which can be widely applied in question parsing and answer sheet correction. however, due to the ambiguities brought by crabbed handwriting and the complicated structure of handwritten mathematics expressions, hmer remains a challenging task. in this paper, we propose a simple and efficient method for handwritten math expression recognition by incorporate semantic graph representation learning into end-to-end training to learn the correlation between different symbols through symbol cooccurrence probabilities. experiments on the crohme dataset and hme100k dataset have validated the effectiveness and efficiency of our method.

Riassunto generato da ProphetNet:
handwritten mathematical expression recognition (hmer) is a challenging task due to the ambiguities brought by crabbed handwritings and handwritten handwritten handwritings. to address these issues, we propose a simple but method to learn the correlation between symbols through symbol cooccurrence probabilities. the proposed method integrates semantic graph representation learning into end - to - end training. to our best knowledge, this method is the first to learn symbol correlation through symbol correlations through symbol representation learning. experiments on the crohme dataset and hme100k dataset have validated the of our method.

Articolo n.3

Testo originale:
mathematical formulas are widely used in online education system, scientific research literature, knowledge questions and answers and other fields. printed mathematical expression recognition (pmer) is to recognize printed mathematical expression image to the corresponding latex sequences. so as to can provide more convenient services for above applications. pmer is different from character recognition. the general steps of character recognition method first cuts a string into several inde-

pendent characters, then extracts features of each character, and finally analyzes and classifies the extracted features to achieve character recognition. pmer is different, since j. long et al. mathematical formulas not only involve segmentation and recognition of mathematical characters, but also need to extract and analyze of two-dimensional structural features of mathematical formulas, it's crucial for the understand of formula semantic information. however, the complexity and diversity of the two-dimensional structure features contained in mathematical formulas are the biggest challenges for pmer. the earliest solution for mathematical expression recognition can be traced back to 1967. anderson et al. proposed a formula structure analysis method based on syntax rules. with the development of this field, some other researchers propose to divide mathematical expression recognition into several sequential execution stages, among which the most in this paper, we propose an improved end-to-end method for printed mathematical formulas image recognition. we use a more advanced image feature extractor to enrich the extracted image features. in order to better realize the representation of the positional relationship between mathematical symbols, we recast the position encoding algorithm proposed in transformer and design a two-dimensional position coding algorithm. in the decoder part, we use bi-gru as core translation module, and we introduce attention mechanism to make translation to focus on the important local information of the feature map generated by encoder. we carry out experiments on the public dataset im2latex-100k, the experimental results show that our method is superior to most of the current advanced methods. acknowledgements. this work is being supported by the national natural science foundation of china under the grant no. u2003208 and no. 62172451, and supported by open research projects of zhejiang lab under the grant no. 2022kg0ab01

Abstract:
printed mathematical expression recognition is to transform printed mathematical formula image into latex sequence. recently, many methods based on deep learning have been proposed to solve this task. however, the positional relationship between mathematical symbols is often ignored or represented insufficient, leading to the loss of structural features of mathematical formulas. to overcome this challenge, we propose a position-aware encoder-decoder model for printed mathematical expression recognition. we design a two-dimensional position encoding algorithm based on sin/cos function to capture positional relationship between mathematical symbols. meanwhile, we adopt a more advanced image feature extraction network. in decoder component, we use bi-gru as the translator, and add attention mechanism to make decoder focus on the important local information. we conduct experiments on the public dataset im2latex-100k, and the results show that our proposed approach is more excellent than the majority of advanced methods.

Riassunto generato da BART:
mathematical formulas are widely used in online education system, scientific research literature, knowledge questions and answers and other fields. printed mathematical expression recognition(pmer) is to recognize printed mathematical expressions image to the corresponding latex sequences. in this paper, we propose a

new end-to-end method for printed mathematical formulas image recognition. we use a more advanced image feature extractor to enrich the extracted image features. in addition, we introduce attention mechanism to focus on the important local information of the feature map generated by the encoder.

Riassunto generato da T5:
mathematical expression recognition (pmer) is widely used to recognize printed mathematical expression images to the corresponding latex sequences. the general steps of pmer involve segmentation and recognition of mathematical expressions, but also need to extract and analyze two-dimensional structural features contained in mathematical formulas. in this paper, we propose an improved end-to-end method for printing mathematical expression image recognition. in order to better realize the representation of the positional relationship between mathematical symbols, we recast the position encoding algorithm proposed in transformer and design a more advanced decoder part, we introduce attention mechanism to focus on the important local information of the feature map generated by encoder. experimental results show that our method is superior to most advanced methods.

Riassunto generato da ProphetNet:
printed mathematical expression recognition (pmer) aims to recognize printed mathematical formulas to the corresponding latex sequences. in this paper, we propose an improved end-to-end method for printed mathematical expressions image recognition. in order to better realize the positional relationship between mathematical symbols, we recast the position encoding algorithm proposed in transformer and design a two-dimensional position coding algorithm. in the decoder part, we introduce attention mechanism to focus on the important local information of the mathematical symbols generated by encoder, and we introduce a translation module to focus attention on the local information associated with each mathematical symbol. experimental results on the public dataset im2latex-100k show that our method is superior to most of the current advanced methods.