Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Architettura di un sistema per l'identificazione di plagi di opere dell'ingegno

Francesco Sergio Pisani, Ettore Ritacco, Danilo Cistaro

RT-ICAR-CS-21-04 **Maggio 2021**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: *www.icar.cnr.it*

Sommario

Introduzione

Obiettivo del documento

Questo documento ha come obiettivo quello di proporre uno strumento per il riconoscimento di un utilizzo non accreditato di opere di ingegno.
Nel campo dell'individuazione di plagi di risorse multimediali, numerosi sistemi sono stati realizzati per rilevare porzioni di testo copiato da altre sorgenti, brani musicali frutto di una evidente operazione di plagio oppure un utilizzo improprio di un'immagine. Nell'ambito del progetto ci si è posto l'obiettivo di definire una metodologia per la realizzazione di un sistema di individuazione dei plagi che possa essere sufficientemente versatile da poter essere applicato a risorse multimediali di natura molto differente: testo, audio, immagini, ecc. Per raggiungere questo obiettivo, una prima analisi dello stato dell'arte è stata effettuata con il duplice obiettivo di stabilire le eccellenze in questo campo e individuare le caratteristiche comuni che un sistema dovrebbe avere per operare in questo ambito. In questo documento verrà mostrato un prototipo in grado di riconoscere dei plagi di porzioni di testo.

Piano di dettaglio dell'attività

Nello specifico verranno trattate le seguenti tematiche.

1. Definizione di un'architettura di riferimento per la realizzazione di un sistema di individuazione di plagi di opere d'ingegno;
2. Realizzazione di un modello di deep learning per l'individuazione di un plagio di un documento di testo;
3. Valutazione delle performance del modello su un dataset di documenti.

Definizioni e acronimi

AI	Artificial Intelligence
API	Application Program Interface
BERT	Bidirectional Encoder Representations from Transformers
LSTM	Long-short term memory
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
RAE	Recurrent Autoencoder
RNNs	Recurrent Neural Networks
SVM	Support Vector Machines
Tf-Idf	Term Frequency - Inverse Document Frequency
UTF	Unicode Transformation Format

Riferimenti

#	Titolo	Note	Link
[1]	El Mostafa, Hambi, and Faouzia Benabbou. "A deep learning based technique for plagiarism detection: a comparative study." IAES International Journal of Artificial Intelligence 9.1 (2020): 81.	-	https://www.researchgate.net/profile/Faouzia_Benabbou/publication/338374695_A_System_for_Ideas_Plagiarism_Detection_State_of_art_and_proposed_approach/links/5f982211458515b7cfa1a23d/A-System-for-Ideas-Plagiarism-Detection-State-of-art-and-proposed-approach.pdf
[2]	Alzahrani, Salha M., Naomie Salim, and Ajith Abraham. "Understanding plagiarism linguistic patterns, textual features, and	-	https://ieeexplore.ieee.org/abstract/document/5766764/

	detection methods." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42.2 (2011): 133-149.		
[3]	Tian, Zhenzhou, et al. "Plagiarism detection of multi-threaded programs via siamese neural networks." IEEE Access 8 (2020): 160802-160814.	-	https://ieeexplore.ieee.org/abstract/document/9184827/
[4]	Barrón-Cedeño, Alberto, et al. "Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection." Computational Linguistics 39.4 (2013): 917-947.	-	https://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00153
[5]	Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).	-	https://arxiv.org/abs/1810.04805
[6]	Feng, Fangxiaoyu, et al. "Language-agnostic bert sentence embedding." arXiv preprint arXiv:2007.01852 (2020).	-	https://arxiv.org/abs/2007.01852
[7]	Kaliyar, Rohit Kumar. "A Multi-layer Bidirectional Transformer Encoder for Pre-trained Word Embedding: A Survey of BERT." 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2020.	-	https://ieeexplore.ieee.org/abstract/document/9058044/
[8]	Canal, Gregory, et al. "Active embedding search via noisy paired comparisons." International Conference on Machine Learning. PMLR, 2019.	-	http://proceedings.mlr.press/v97/canal19a.html
[9]	Kłusek, Adrian, and Witold Dzwinel. "Multi-GPU k-Nearest Neighbor search in the context of data embedding." Advances in Parallel Computing 32 (2018): 359-368.	-	https://books.google.it/books?hl=it&lr=&id=ysFVDwAAQBAJ&oi=fnd&pg=PA359&dq=nearest+neighbor+search+embedding&ots=k65wPmb74u&sig=XdnLLpgU2Mi7lxKLd3qOudmR86k
[10]	Zhang, Xiyuan, Yang Yuan, and Piotr Indyk. "Neural embeddings for nearest neighbor search under edit distance." (2019).	-	https://openreview.net/forum?id=HJlWIANtPH
[11]	Vani, K., and Deepa Gupta. "Identifying document-level text plagiarism: A two-phase approach." Journal of Engineering Science & Technology (JESTEC) 12.12 (2017): 3226-3250.	-	http://jestec.taylors.edu.my/Vol%2012%20issue%2012%20December%202017/12_12_9.pdf
[12]	Al-Bayed, Mohran H., and Samy S. Abu-Naser. "Intelligent Multi-Language Plagiarism Detection System." (2018).	-	http://dstore.alazhar.edu.ps/xmlui/handle/123456789/125
[13]	BBC Full Text Document Classification	-	https://www.kaggle.com/shivamkushwaha/bbc-full-text-document-classification
[14]	DistilBERT base model (uncased)	-	https://huggingface.co/distilbert-base-uncased

Algoritmo di individuazione di plagi

Nel campo dell'individuazione di plagi di risorse multimediali, numerosi sistemi sono stati realizzati per rilevare porzioni di testo copiato da altre sorgenti, brani musicali frutto di una evidente operazione di plagio oppure un utilizzo improprio di un'immagine. Nell'ambito del progetto ci si è posto l'obiettivo di definire una metodologia per la realizzazione di un sistema di individuazione dei plagi che possa essere sufficientemente versatile da poter essere applicato a risorse multimediali di natura molto differente: testo, audio, immagini, ecc. Per raggiungere questo obiettivo, una prima analisi dello stato dell'arte è stata effettuata con il duplice obiettivo di stabilire le eccellenze in questo campo e individuare le caratteristiche comuni che un sistema dovrebbe avere per operare in questo ambito.

Il risultato di questo lavoro è stato quello di definire un'architettura di riferimento che, con le opportune modifiche e personalizzazioni, possa essere applicabile sia al testo che all'audio oppure alle immagini. Nel proseguo di questo documento verrà descritta un'implementazione riferita al caso dell'analisi di documenti testuali.

L'architettura di riferimento può essere generalizzata dalla Figura 1. Il processo di identificazione di plagi può essere schematizzato nei seguenti sotto-processi:

- Segmentazione
- Embedding
- Confronto con una base di dati

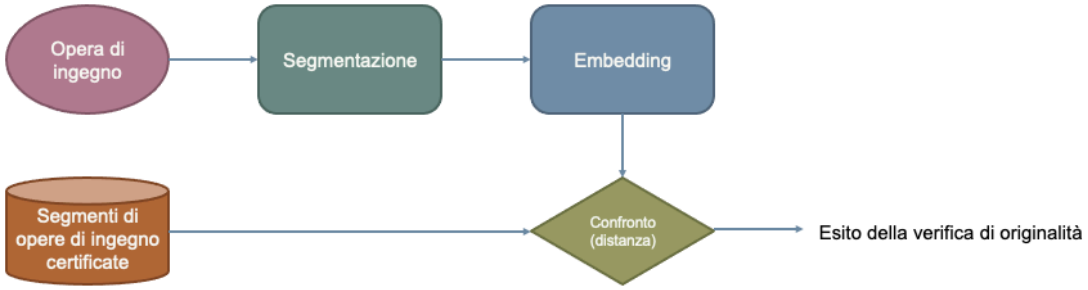

Figura 1 - Esempio di Architettura

Il processo inizia analizzando un'opera di ingegno che viene suddivisa in sotto blocchi. Ogni blocco contiene una porzione semanticamente autosufficiente dell'opera. Ad esempio, un documento potrebbe essere un suddiviso in blocchi con un numero prefissato di parole. Per una risorsa audio potrebbero essere blocchi di alcuni secondi. Questa fase del processo viene indicata come *segmentazione* dell'opera.

Il passo successivo consiste nel calcolare un *hash*, cioè un codice numerico che permette di identificare univocamente un blocco nel dominio di riferimento. Ovviamente non è possibile assegnare una semantica a codici hash calcolati in domini differenti, cioè non è possibile comparare un hash calcolato su un blocco testuale con uno calcolato su un segmento audio. Questa fase viene indicata come calcolo del codice di *embedding* perché il codice numerico ottenuto gode di una proprietà: blocchi "simili" corrispondono a valori di embedding "vicini". Nel

resto del documento i termini embedding, vettore di embedding, codice hash saranno utilizzati come sinonimi per indicare il vettore numerico che rappresenta un singolo blocco di testo.

Il concetto di *simile* dipende dal dominio e corrisponde al concetto di uguaglianza. Ad esempio due testi contenenti le stesse parole con una sola parola differente vengono considerati *praticamente* uguali, perciò simili. Allo stesso modo, due valori di embedding si considerano *vicini* quando, secondo una funzione di distanza, la differenza tra i due valore è inferiore ad una certa soglia.

Dopo aver calcolato il codice *hash* per un blocco, si effettua la ricerca in una base di dati nota in cui sono stati registrati i codici di tutti i blocchi per tutte le opere di cui si voglia identificare eventuali plagi. La ricerca avviene confrontando i valori di hash: se non vengono trovati risultati, allora il blocco può essere considerato come un'opera originale. Ovviamente il processo di confronto rispetto ai valori memorizzati in una base di dati potrebbe restituire più di un risultato. Questo perché esiste sempre un margine di errore nel processo di calcolo del valore di hashing e si riflette nell'individuazione di falsi positivi: opere che vengono rilevate come copie di altre opere ma che in realtà sono opere originali. Minimizzare il numero di falsi positivi dipende dalla qualità dell'algoritmo di calcolo del vettore di embedding e sarà una delle metriche di valutazione delle performance dell'algoritmo.

Uno dei limiti del sistema è la costituzione della base di dati delle opere note. Per poter identificare delle opere come plagio, è necessario avere dei riferimenti. Quindi il sistema potrà individuare solo le opere presenti nella base di dati.

Poiché l'obiettivo è la realizzazione di un sistema per l'identificazione dei plagi, questo documento presenterà un caso applicativo applicato all'analisi di documenti testuali.

Architettura

Per sviluppare un sistema di identificazione di plagi di documenti testuali occorre effettuare delle scelte architetturali per poter analizzare dei testi. Ogni componente logico dell'architettura di riferimento (Figura 1) deve essere sostituito con un componente in grado di espletare la relativa funzione.

Per ogni componente sono state individuate delle soluzioni tecnologiche che rappresentano lo stato dell'arte nel proprio ambito. Poi è stato creato un algoritmo di calcolo dell'hash di un blocco di testo.

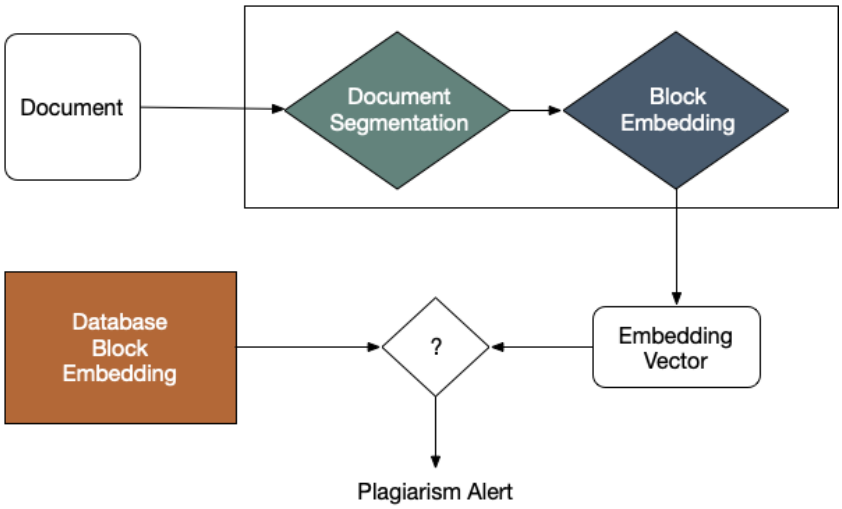

Figura 2 – Architettura per l'analisi del testo

Nella Figura 2 sono raffigurati i componenti che compongono l'architettura del prototipo realizzato. L'input è rappresentato da un insieme di documenti testuali e da una base di dati di vettori di embedding. La base di dati contiene i valori di embedding calcolati sui blocchi di tutti i documenti che si vuole monitorare, cioè per i quali si vuole individuare eventuali plagi.

Il modulo di calcolo dell'embedding è composto da due sotto-moduli: il primo suddivide un documento in blocchi di dimensione prefissata, il secondo effettua il calcolo del vettore di embedding per ogni blocco.

Dopo aver calcolato un embedding, il sistema effettua una ricerca sul database di embedding noti. Se la ricerca ha esito positivo, allora si è in presenza di un plagio. Il processo di uguaglianza tra due embedding è basato sul concetto di distanza. In questo modo se la distanza tra due valori è inferiore ad una certa soglia, i due embedding si considerano uguali e quindi si è in presenza di un plagio. Il valore ottimale della soglia è stimato durante la fase sperimentale.

Nei prossimi capitoli saranno esaminati in dettaglio le singole componenti del sistema.

Descrizione algoritmo

Il componente che calcola il valore di embedding per un blocco di testo è costituito da una rete neurale che estende un algoritmo noto in letteratura come BERT. Per costruire un modello efficace di calcolo dell'embedding è stato utilizzata una struttura chiamata rete siamese. Nella Figura 3 è riportato uno schema dell'architettura. Nella fase di addestramento la rete riceve due input: due blocchi testuali di cui si vuole calcolare l'embedding e di cui si conosce l'etichetta, cioè se i due blocchi sono simili (plagio) oppure dissimili. Una rete neurale basata su BERT calcola un vettore di embedding per ogni input. Per addestrare la rete a calcolare vettori di embedding vicini per testi simili, si utilizza una funzione di distanza. Nella figura viene utilizzata la funzione cosine similarity, tale funzione restituisce 1 quando due input A e B sono uguali, -1 quando sono opposti e 0 quando sono dissimili. Questa funzione è legata al concetto di distanza dalla relazione:

$$dist(A, B) = 1 - \mathrm{sim}(\mathrm{A}, \mathrm{B})$$

in questo modo è possibile trasformare il processo di ottimizzazione della rete in un processo di ottimizzazione della similarità fra due vettori.

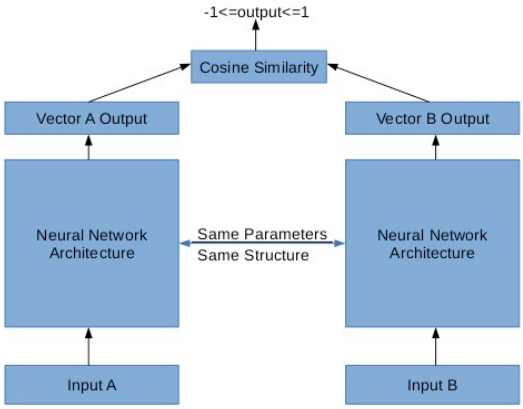

Figura 3 - Struttura delle rete DNN

Sfruttando la relazione tra distanza e similarità, il processo di addestramento si semplifica perché è sufficiente sfruttare il concetto di similarità del coseno e definire la seguente funzione di loss:

$$\text{loss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases}$$

Dove *margin* è una costante di regolarizzazione e x_1, x_2 sono i blocchi di testo in input e y è l'etichetta che indica se x_1 e x_2 sono simili (1), opposti (-1) oppure completamente diverse (0). La funzione di loss è la funzione che il processo di ottimizzazione della rete tenta di minimizzare. Ad un valore tendente a zero corrisponde la configurazione di pesi ottimale.
Nella fase di valutazione, il modulo di calcolo restituirà soltanto il vettore di embedding. Questo valore sarà utilizzato per cercare eventuali blocchi precedentemente inseriti nella base di dati e, in tal caso, si rileva un potenziale plagio.
La fase di ricerca di un embedding consiste nel cercare tutti i valori la cui distanza è inferiore ad una cera soglia: se non vengono trovati risultati, allora non è stato individuato un plagio. Questo metodo si basa sul principio che due vettori di embedding sono tanto più vicini, cioè la distanza è tendente a zero, quanto più sono simili i due blocchi sui quali sono stati calcolati. Il processo di addestramento con la funzione di loss definita in precedenza hanno lo scopo di definire un modello di deep learning che produce vettori di embedding con queste caratteristiche. Il valore ottimale della soglia sarà scelto nella fase sperimentale di addestramento del modello.
Nel capitolo successivo sarà presentato un approfondimento sul funzionamento dell'algoritmo BERT.

Descrizione del modulo BERT

BERT (Bidirectional Encoder Representations from Transformers) è un algoritmo ideato da Google per l'analisi di testi. Si tratta di un natural language processor in grado di analizzare una frase all'interno di un contesto di riferimento e quindi superare uno dei limiti maggiori degli algoritmi tradizionali per l'analisi del testo: quello di non riuscire a distinguere frasi il cui significato dipende dal contesto. Questo aspetto risulta estremamente significativo nell'ambito in cui è stato utilizzato per primo questo algoritmo, l'analisi di query di ricerca. Fornendo un meccanismo per analizzare in maniera più efficace query scritte in linguaggio

naturale, BERT si è rivelato molto più efficace dei sistemi precedenti. Numerose estensioni di BERT sono state realizzate e oggi questo algoritmo può essere utilizzato in numerosi ambiti:

- Sentence embedding
- Clustering
- Text Mining
- Sentence translation
- Semantic search
- Query answering
- Text generation
- Text summarization
- Ecc.

L'innovazione principale rispetto agli algoritmi precedenti risiede nella capacità del modello di considerare i termini in entrambe le direzioni, ovvero le parole presenti prima e dopo una data parola, nella fase di training dell'algoritmo. Questa tecnica viene sfruttata per realizzare i due step principali dell'algoritmo. Per comprendere meglio il funzionamento, consideriamo la frase "*The man goes to the store. He buys a Gallon of Milk*".
L'algoritmo esegue due operazioni. La prima è quella di *Word Masking*. Prima di codificare la frase come input del modello, il 15% delle parole in ogni sequenza viene nascosto (o mascherato). Poi si tenta di predire il valore originale della parola mascherata basandosi sul contesto fornito dalle altre parole nella sequenza (in entrambe le direzioni). Nell'esempio:

"The man goes to the [Mask1]. He buys a [Mask2] of Milk".

L'algoritmo cerca di predire [Mask1] come "store" e [Mask2] come "gallon". Questo meccanismo permette di apprendere la relazione tra parole in una frase.
La secondo operazione è quella di *Prediction*. In questa fase si cerca di apprendere le relazioni che intercorrono tra frasi. Cioè se la frase B segue la frase A o viceversa. I due esempi seguenti mostrano le deu possibili relazioni per la frase di esempio precedente:

Frase A: "*The man goes to the* Store*"*
Frase B: *"He buys a Gallon of Milk"*
Label: IsNextSentence

Frase A*: "The man goes to the* Store
Frase B*: "Penguins are Flightless*"
Label : NotNextSentence

Nel primo esempio il modello apprende come molto probabile che la frase B possa venire dopo la frase A e la etichetta come "IsNextSentence", mentre riconosce come molto improbabile che si verifichi il secondo esempio. Ciò consente a BERT di imparare quale frase può seguire un'altra. Questa tecnica di training viene eseguita su grandi corpus di dati. BERT usa l'intero corpus di wikipedia per costruire il modello.
Altri elementi chiave del successo di BERT sono l'applicazione del training bidirezionale del transformer e il meccanismo dell'*attention*, molto utilizzato nella modellazione del linguaggio. Il concetto di *attention* si utilizza per "focalizzare" l'attenzione del modello su particolari aspetti dell'input. In BERT l'attention viene utilizzata per "pesare" l'importanza dei singoli token nel task di riferimento.

I risultati ottenuti mostrano che un modello linguistico addestrato in modo bidirezionale può abilitare alla comprensione della semantica, del contesto e del flusso. L'addestramento bidirezionale è reso possibile tramite una tecnica denominata Masked LM (MLM).
Un altro aspetto che risulta importante per la comprensione delle capacità di BERT è il concetto di *transfer learning*. Nel campo della visione artificiale, ad esempio, è stato più volte dimostrato il valore del transfer learning che consiste nel pre-addestramento di un modello di rete neurale su un task o dominio e successivamente, tramite *fine-tuning*, l'utilizzo dello stesso modello addestrato come base di un nuovo modello *purpose-specific* (eventualmente appartenente anche ad un dominio differente). Lo stesso approccio può essere proficuamente impiegato in molte attività di NLP. In BERT questa tecninca è ampiamente impiegata e utilizzata insieme a quella denominata *feature-based training*. In questo approccio, una rete neurale pre-addestrata produce *word embeddings* che vengono impiegati come feature nei modelli NLP più complessi.
Queste tecniche e queste caratteristiche vengono sfruttate per costruire la parte più importante di BERT: l'encoder. L'input è una sequenza di token estratti dalla frase da analizzare. L'output è una sequenza di numeri che rappresenta l'encoding dei singoli token.
Per migliorare la capacità del modello di comprendere il modello, l'apprendimento bidirezionale è utilizzato insieme alla tecnica *Masked LM (MLM)*. Prima di inserire le sequenze di parole in BERT, il 15% delle parole in ciascuna sequenza viene sostituito con un token *[MASK]*. Il modello tenta quindi di prevedere il valore originale delle parole mascherate, in base al contesto fornito dalle altre parole non mascherate nella sequenza. In pratica la predizione del token mancante si ottiene calcolando una probabilità per ogni parola del vocabolario costruito a partire dal dataset di addestramento e la funzione di loss di Bert prende in considerazione solo i valori mascherati ignorando la predizione dei valori non mascherati. In conseguenza di ciò, il modello converge più lentamente rispetto ai modelli direzionali, una caratteristica che è compensata dalla sua maggiore awareness del contesto.
Per ottenere la capacità di predizione e comprensione del testo, BERT combina diverse strategie. La *Next Sentence Prediction* (NSP) è una di queste. Nel processo di training, il modello riceve coppie di frasi come input e impara a predire se la seconda frase nella coppia è la frase successiva nel documento originale. Durante il training, il 50% degli input sono una coppia in cui la seconda frase è la frase successiva nel documento originale, mentre nell'altro 50% viene scelta come seconda frase una frase casuale dal corpus. Per raggiungere questo scopo, l'input viene pre-elaborato nel modo seguente:

1. Un token [CLS] viene inserito all'inizio della prima frase e un token [SEP] viene inserito alla fine di ogni frase.
2. A ogni token viene aggiunta una frase target.
3. Un'informazione posizionale viene aggiunta a ciascun token per indicarne la posizione nella sequenza.

Per capire se la seconda frase è effettivamente collegata alla prima, vengono eseguiti i seguenti step:

1. L'intera sequenza di input passa attraverso il modello.
2. L'output del token [CLS] viene trasformato in un vettore 2 × 1, utilizzando un semplice layer denso.
3. Viene calcolata la probabilità di IsNextSequence (ossia il prossimo elemento della sequenza) tramite una funzione di attivazione softmax.

Nella fase di training del modello di BERT, Masked LM e Next Sentence Prediction vengono addestrati insieme, con l'obiettivo di minimizzare la funzione di loss combinata.

Fine-tuning

L'utilizzo di BERT per un task specifico è relativamente semplice. Il modello viene esteso aggiungendo ulteriori layer e riutilizzando i moduli principali pre-addestrati, cioè ereditando la configurazione dei parametri stimati in precedenza. Tramite questo approccio, è possibile personalizzare e impiegare BERT in un'ampia varietà di task linguistici:

- Task di classificazione (ad es. sentiment analysis) sono eseguiti in modo simile alla Next Sentence classification, aggiungendo un layer di classificazione dopo l'output del transformer per il token [CLS].
- In task di Question Answering, il software riceve una domanda riguardante una sequenza di testo e viene richiesto di contrassegnare la risposta nella sequenza. Utilizzando BERT, un modello di Question Answering può essere addestrato apprendendo due vettori aggiuntivi che identifichino l'inizio e la fine della risposta.
- Nel caso della Named Entity Recognition (NER), il software riceve una sequenza di testo ed è richiesto di contrassegnare i vari tipi di entità (persona, organizzazione, data, ecc.) che appaiono nel testo. Utilizzando BERT, un modello NER può essere addestrato inserendo il vettore di output di ciascun token in un layer di classificazione che predice l'etichetta NER.

Word Masking

L'addestramento del modello linguistico in BERT viene effettuato predicendo il 15% dei token nell'input, scelti casualmente. Questi token vengono pre-elaborati come segue: l'80% viene sostituito con un token "[MASK]", il 10% con una parola casuale e il 10% utilizza la parola originale. L'intuizione che ha portato gli autori a scegliere questo approccio è la seguente:

- Usando [MASK] per il 100% delle volte, il modello non produrrebbe necessariamente buone rappresentazioni simboliche per parole non mascherate. I token non mascherati sarebbero ancora utilizzati per il contesto, ma il modello sarebbe ottimizzato per predire parole mascherate.
- Usando [MASK] per il 90% delle volte e parole casuali per il 10% delle volte, questo insegnerebbe al modello che la parola osservata non è mai corretta.
- Usando [MASK] per il 90% delle volte e mantenessimo la stessa parola il 10% delle volte, il modello potrebbe semplicemente copiare l'embedding non contestuale.

Architettura

A livello architetturale, BERT è uno stack di transformer encoder layers, che consistono in self-attention "heads" multiple. Cioè ogni livello calcola degli encoding dell'input tramite blocchi attention-based. Per ogni token di input in una sequenza, ogni head calcola i vettori di chiavi, valori e query, usati per creare una rappresentazione pesata. Gli output di tutte le head nello stesso layer vengono combinati e propagati ad un layer fully-connected. Ogni livello è connesso tramite una skip connection e seguito da un livello di Batch Normalization. Il flusso di funzionamento di BERT consiste di due fasi principali: il pre-training e il fine-tuning. Il pre-training usa due task supervisionati: l'utilizzo di tecniche di "masking" (MLM, predizione di token in input "mascherati") e la predizione della frase successiva (NSP, acronimo di next sentence prediction, ovvero predire se due frasi in input sono tra loro adiacenti). Le rappresentazioni dell'input sono calcolate come segue: ogni parola nell'input viene prima tokenizzata in segmenti di parole, e successivamente sono combinati tre embedding layer (token, position, e segment), al fine di ottenere un vettore di lunghezza fissata. Esistono numerose varianti di BERT, inclusa la versione originale "base" e varianti "large". Esse variano nel numero di heads layer e dimensione degli strati interni.

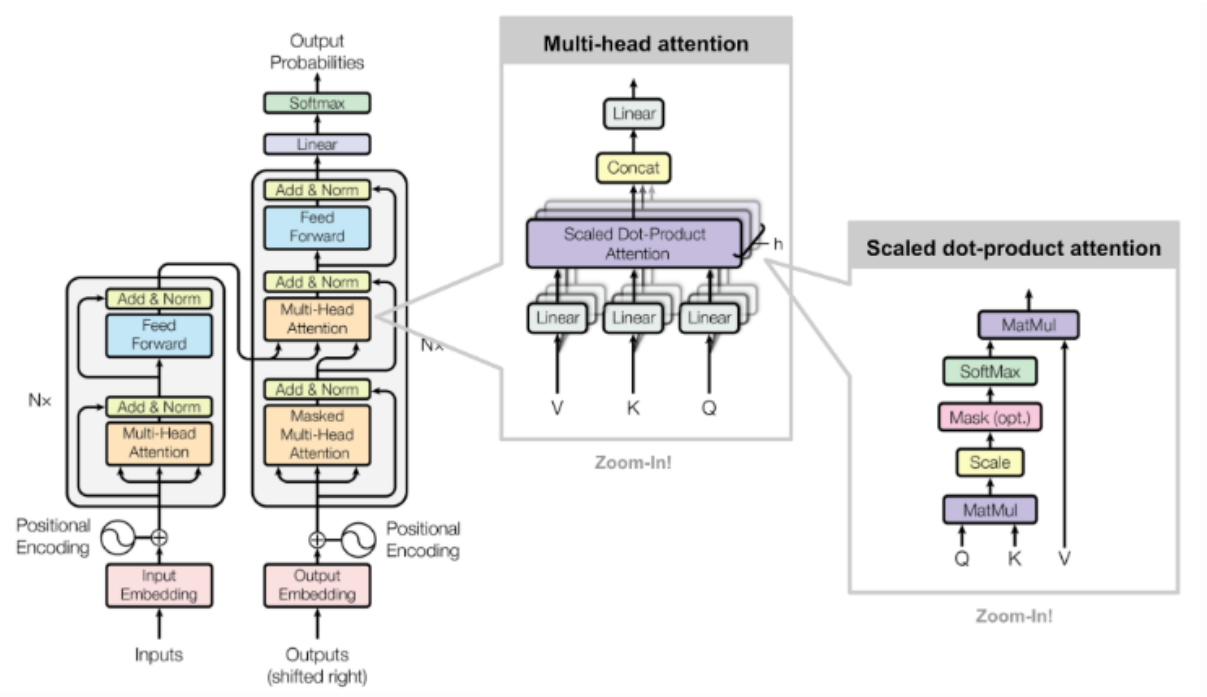

Figura 4 - Architettura di BERT

Protocollo di sperimentazione

L'algoritmo presentato nei capitoli precedenti è stato utilizzato per realizzare un prototipo e misurarne le performance su un dataset di documenti testuali.

Nei prossimi paragrafi verranno descritti il protocollo di valutazione e i risultati ottenuti.

Dataset

Per misurare le perfomance dell'algoritmo è stato scelto un dataset costituito da 2225 documenti contenenti le news pubblicate sul portale della BBC e suddivisi in 5 categorie: business, entertainment, politics, sport, tech. Ogni documento è costituito dal titolo della news e dal relativo testo.

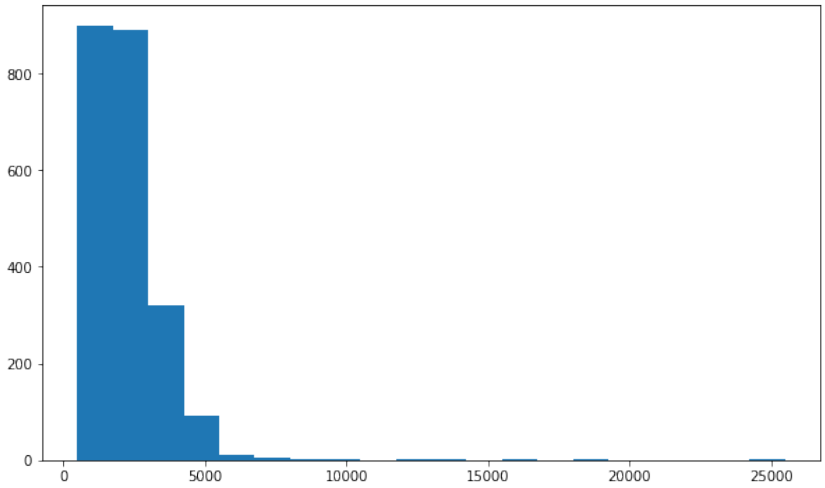

Figura 5 - Dimensione dei documenti in numero caratteri

Nella Figura 5 è possibile vedere la distribuzione dei documenti in termini di lunghezza misurata in numero di caratteri. La maggior parte delle notizie contiene tra i 500 e i 5000 caratteri, con pochi esempi di testi molto lunghi.

Per ottenere un dataset compatibile con l'algoritmo proposto, ogni documento è stato suddiviso in blocchi da 256 token. Per token si intende la singola parola del testo. Ogni blocco è ottenuto analizzando il documento con una finestra scorrevole tale per cui due blocchi consecutivi condividono una porzione del testo, nella sperimentazione è stato utilizzando un valore del 30%. Questo valore permette di simulare documenti in cui piccole porzioni di testo in comune non indicano l'esistenza di un plagio.

Per simulare la presenza di blocchi di plagio e permettere al modello di apprendere un modello efficace, una porzione pari al 10% circa del dataset è stata utilizzata per simulare un blocco come plagio in accordo ad una delle seguenti regole:

- Si sostituiscono il 10% delle parole di un blocco con parole casuali appartenenti al vocabolario del dataset.
- Si costruisce il blocco di plagio come un blocco che condivide il 70% del testo del blocco originale.

Il dataset utilizzato per la sperimentazione è stato costruito generando delle coppie di blocchi di testo, alternando coppie di blocchi "non simili" con etichetta "0" a coppie di blocchi "simili", quindi che rappresentano un plagio, con etichetta "1" in un rapporto di 10 a 1. La numerosità tra coppie "non plagio" e coppie "con plagio" è mostrata nella Figura 6.

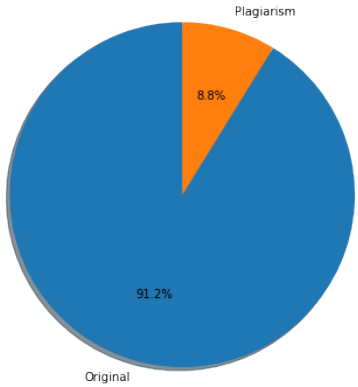

Figura 6 - Dataset BBC

Modello

Il modello proposto implementa l'algoritmo presentato nei capitoli precedenti. L'implementazione è basata sul framework pytorch e un modello di BERT pre-addestrato su un dataset estratto dalla versione inglese di Wikipedia.

Il modello finale è costituito da transformer BERT e da un layer di mean pooling che produce un vettore di emebedding di dimensione 768. Quindi per ogni blocco di testo di dimensione pari a 256 token viene generato un vettore di 768 numeri float. La rete di addestramento è costituita da due reti gemelle che condividono la configurazione dei parametri e i vettori di embedding prodotti vengono confrontati per calcolarne la similarità.

La funzione di loss utilizzata è la CosineSimilarity che calcola la funzione di similarità del coseno introdotta nei capitoli precedenti. Il dataset in input è stato suddiviso in tre partizioni: train, validation e test con le seguenti proporzioni rispetto al dataset originale 55%, 15 % e 30%.

Il modello è stato addestrato sul dataset di train per 10 epoche. Per ogni epoca sono state misurate le performance utilizzando il coefficiente di correlazione di Pearson. Questa metrica restituisce una misura di correlazione tra ciò che ha appreso il modello sulle coppie di blocchi e l'etichetta di classe associata alla coppia (plagio/non plagio). Il modello con valore massimo del coefficiente di Pearson ottenuto sul validation dataset è stato scelto come modello di riferimento e con quello sono state valutate le performance sul dataset di test. Ovviamente, i dati contenuti nel dataset di validation e test non sono stati mai utilizzate per addestrare il modello.

Risultati sperimentali

Utilizzando il modello e il protocollo presentati nel capitolo precedente, sono stati analizzati i risultati sul test set.

Il protocollo di valutazione prevede di calcolare il codice hash di ogni blocco, misurare la distanza tra i codici hash di due blocchi (o in maniera equivalente calcolare il valore della similarità del coseno) e assegnare un'etichetta 0/1 per indicare se nella coppia di blocchi (A, B), il blocco B può essere considerato un plagio del blocco A. Il confronto con l'etichetta reale presente nel dataset permette di capire se il modello ha classificato correttamente la coppia. In questo capitolo verranno presentati i risultati ottenuti valutandoli attraverso alcune metriche di riferimento.

Metrica	Pearson
Cosine-Similarity	0.9457
Manhattan-Distance	0.9387
Euclidean-Distance	0.9406
Dot-Product-Similarity	0.9290

Nella tabella precedente sono riportati i risultati ottenuti. Il valore del coefficiente misura la qualità del valore di embedding calcolato dal modello. Un valore pari a 1, significa ottenere dei vettori che riproducono un caso ideale in cui a blocchi simili corrispondono vettori con distanza quasi nulle e a blocchi dissimili corrispondono vettori con distanza elevata. I valori ottenuti permettono di affermare che il modello calcola dei vettori di "buona" qualità. Ma per comprendere meglio la capacità del modello di identificare i plagi, sono state calcolate alcune metriche tipiche dei task di classificazione. In questo modo, scegliendo una soglia in base alla quale convertire la misura di distanza tra due vettori di embedding in etichetta 0 (distanza maggiore della soglia) oppure 1 (distanza minore della soglia), è possibile individuare coppie che indicano plagi da coppie che non lo sono e avere un'indicazione maggiormente comprensibile rispetto al task di riferimento.

Nella tabella seguente vengono riportati i risultati delle seguenti metriche calcolate utilizzando il metodo del coseno per calcolare la distanza tra i due vettori di embedding:

1. Accuratezza, che restituisce il rapporto tra il numero di predizioni corrette rispetto al numero totale di coppie nel dataset;

2. Precisione, che indica quanto il modello è preciso nell'individuare i plagi;
3. Recall, che indica quanti sono i plagi individuati rispetto al totale dei plagi presenti nel dataset;
4. F1 score, che rappresenta la media armonica tra precisione e recall;
5. Precisione media, che calcola la media della precisione calcolata separatamente sulla classe 0 e la classe 1;

Metrica	Valore
Accuratezza	99.78
Precisione	98.70
Recall	98.76
F1-score	98.73
Precisione Media	99.67

I valori ottenuti dimostrano come il modello riesce a distinguere in maniera efficace le coppie con plagio dalle coppie senza. Ma per individuare il valore di soglia ottimale e comprendere come questa scelta possa influenzare la capacità del modello di individuare correttamente i plagi, sono stati calcolati due grafici. Il primo rappresenta il grafico della curva di ROC, che mette a confronto il valore del numero di falsi positivi e il valore dei veri positivi al variare del valore della soglia. La linea tratteggiata di colore rosso rappresenta un classificatore random, e l'area sottesa alla curva rappresenta numericamente quanto la classificazione è corretta; un valore pari a 1 indica un classificatore ottimo.

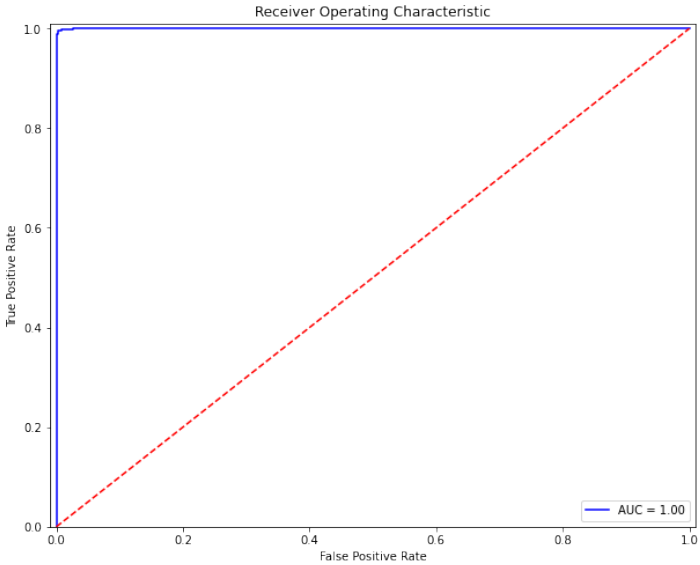

Figura 7 - Curva di ROC

Il secondo grafico presenta la curva tra precisione e recall e permette di capire come la scelta della soglia può influire su queste due metriche. Anche in questo caso la curva tratteggiata rappresenta un classificatore random e il valore di precisione media (AP) pari a 1 indica il valore ottimo.

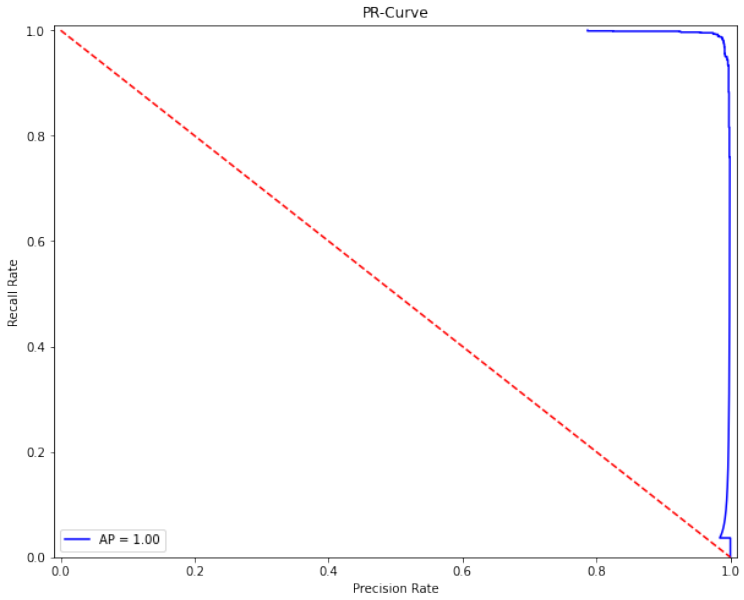

Figura 8 - Curva precisione vs recall

Sistema di ricerca efficiente di plagi

Il modello presentato nei capitoli precedenti permette di calcolare un codice hash da confrontare con una base di dati di codice precedentemente calcolati. Per effettuare una ricerca efficiente è possibile utilizzare un vector database ottimizzato per questo scopo.

Un database vettoriale è progettato per gestire enormi milioni, miliardi o addirittura trilioni di set di dati vettoriali. Questa tipologia di database può essere utilizzata in molteplici ambiti come la visione artificiale, i motori di raccomandazione, ecc.

Il concetto di vettore permette di definire un'entità da memorizzare nel database per poi effettuarne una ricerca in funzione di una qualche funzione di similarità o distanza. Tra i meccanismi più comuni per ottimizzare la memorizzazione dei dati, molti utilizzano il concetto di partizione: i dati vengono suddivisi temporalmente e le query vengono eseguite parallelamente su tutte le partizioni. Per query si intende cercare i k vettori più simili rispetto al vettore fornito come parametro. Per questo motivo, un database vettoriale deve supportare le metriche di distanza più comuni, quali la distanza euclidea, la distanza di manhattan, ecc.

Un'altra caratteristica che un database vettoriale dovrebbe avere è il supporto all'elaborazione distribuita. Oltre alla numerosità dei dati, bisogna considerare anche il numero di richieste che il servizio dovrà erogare. Una soluzione distribuita permette di bilanciare il carico su più nodi e automaticamente permette di scalare nel numero di richieste che il servizio riesce a gestire.

La ricerca dei vettori degli embedding è un elemento cruciale in questo applicativo, poiché per ogni test di plagio è necessario confrontare il codice di hash con tutti i codici della base di dati pre-costituita. Quindi una ricerca efficiente è un elemento fondamentale affinché il sistema possa risultare concretamente realizzabile.

Conclusioni

il problema affrontato in questa attività è l'uso improprio di opere di ingegno. In particolare, è stato raggiunto l'obiettivo di realizzare un sistema che sia in grado di individuare in maniera efficiente eventuali plagi di documenti. Nel corso dell'attività è stata definita un'architettura di riferimento che si può qualificare come metodologia da applicare sia a documenti testuali sia a risorse multimediali di natura differente, quali audio, immagini, ecc. Questa architettura è stata usata per realizzare un prototipo che analizza blocchi di testo per individuare eventuali similitudini con documenti memorizzati in una base di dati contenenti opere da preservare rispetto ad utilizzi illegittimi.

L'efficacia della soluzione proposta è stata valutata tramite una sperimentazione condotta utilizzando un dataset pubblico di documenti testuali. I risultati ottenuti hanno dimostrato come il sistema sia in grado di individuare efficacemente eventuali plagi.