Progetto Pos-Tagging Contaldi S., Russano A.

Abstract

Nel corso di questo progetto è stato richiesto di adattare lo script per il *token classification* proposto da *Hugging Face Transformer* per la *Named Entity Recognition* (NER) al *Pos Tagging* per l'italiano.
Al termine del progetto, i risultati del codice sviluppato sono stati confrontati con un opportuno benchmark di riferimento.

1. Introduzione

Il progetto vede lo sviluppo di un codice per il *Pos Tagging* a partire da uno per la *Named Entity Recognition* (da ora NER).

La **NER** si occupa di indicare, all'interno di un testo selezionato, tutti i token al suo interno che possono essere associati a nomi specifici (es: nomi di persona, luoghi, organizzazioni, date, categorie predefinite…).

Il **Pos Tagging** si occupa di indicare, per ogni token del testo selezionato, l'etichetta (il tag) di riferimento: ovvero, la categoria grammaticale di appartenenza in un contesto linguistico specifico e in una data varietà.

All'interno della cartella fornita, sono stati resi disponibili diversi file per la resa del task richiesto.

- Dataset su cui eseguire il *training* del proprio codice adattato con possibilità di scelta tra due diversi sistemi di annotazione (Eagles e Distrib);
- file .py contente il codice per la NER su cui lavorare;
- Dataset per il *testing* dell'apprendimento del modello proveniente dalla campagna internazionale EVALITA 2007.

Per l'esecuzione del task richiesto sono state utilizzate tecniche di *Deep Learning* specifiche per l'NLP, come l'utilizzo del modello pre-addestrato di BERT: un *Large Language Model* (LLM) sviluppato da Google.

Il progetto è stato eseguito utilizzando i servizi di Google Colab.

1.1 Architettura adottata

Nell'impostare il progetto, come prima cosa è stato scelto il modello pre-addestrato di riferimento su cui eseguire il codice: *sachaarbonel/bert-italian-cased-finetuned-pos* (link di riferimento: https://huggingface.co/sachaarbonel/bert-italian-cased-finetuned-pos). La scelta è ricaduta su questo modello per le sue specifiche tecniche:

- è "*italian cased*", dunque mantiene la differenza tra maiuscole e minuscole;
- ha una struttura potente (12 layer e 110 M parametri);
- è addestrato su un corpus etichettato per i tag grammaticali; dunque, non è un modello BERT generico ma è *fine-tuned* per il *Pos Tagging*;
- è consigliato per la lingua italiana.

In secondo luogo, è stato selezionato il Dataset su cui effettuare il *training* e *validation*. Si è optato per il sistema di annotazione "Eagles", in quanto dotato di una maggiore eterogeneità tra le etichette proposte: questo si è rivelato funzionale durante la conversione dei tag del dataset a quelli del modello in formato *Universal Pos Tag Set* (UPOS) .

Inoltre, sono stati installati i pacchetti necessari per il funzionamento del codice: "scikit-learn" per il calcolo delle metriche ed "evaluate" per la valutazione delle metriche.

Infine, all'interno della riga di comando, sono state indicate informazioni necessarie per il training del modello che richiamano gli argomenti del codice.

2. Sviluppo del codice

Lo sviluppo del codice, nell'adattamento da NER a *Pos Tagging*, ha richiesto diverse fasi:

2.1 Adattamento etichette

Innanzitutto, è stato necessario adattare le etichette del Dataset di riferimento alle etichette UPOS riconosciute dal modello selezionato per il progetto. E' stato, dunque, creato un dizionario che convertisse i tag Eagles in tag UPOS affinché il modello potesse leggere le etichette e riconoscerle adeguatamente. Inoltre, è stata definita una funzione "map_custom_to_upos" che permette di mappare un tag Eagles al corrisponde UPOS prendendo solo il primo carattere del tag, definendone la categoria principale.

2.2 Data processing

La procedura di *data processing* ha richiesto il caricamento e la lavorazione del Dataset selezionato per renderlo congruo e utilizzabile al codice e al task richiesto.

Il Dataset è stato caricato all'interno del codice grazie al pacchetto Pandas, rinominando le colonne in "token" e "tag" e pulendo il Dataset da eventuali tag speciali e spazi vuoti. È stato, inoltre, effettuato lo *shuffle* dei dati.

Successivamente, il Dataset è stato suddiviso per il *training* (80% del totale) e *validation* (20% del totale).

Una volta diviso, il Dataset è stato accuratamente tokenizzato grazie al modello scelto per il *Pos Tagging*: a ogni label è stato associato uno specifico ID di riferimento ed è stata indicata la lunghezza massima di riferimento per la tokenizzazione. I token ottenuti sono stati associati ai tag di riferimento.

Infine, i Dataframe sono stati mappati e convertiti in Dataset *Hugging Face*.

2.3 Ulteriori modifiche apportate

Dopo il lavoro di *data processing* (spiegato al paragrafo 2.2) che ha visto una modifica del codice di partenza per un adattamento al task richiesto, sono stati apportati ulteriori cambiamenti necessari per rendere il codice congruo alla consegna.

Sono stati stabiliti i *dataloader* necessari per gestire il processo di *padding* e *batching* in modo coerente con gli input e i tag prefissati, addestrare il modello con *shuffle* dei dati e validare il modello a fine *training*.

Infine, sono state adattate le metriche per renderle idonee al task (per maggiori informazioni si rimanda al paragrafo 3) ed è stato stabilito un file .json in cui salvare i risultati ottenuti.

3. Metriche e risultati ottenuti

Per definire le metriche adeguate al task, è stato necessario sostituire il pacchetto di riferimento. Si è passati dal pacchetto "seqeval", tipico della NER, a quello di "evaluate" più idoneo al Pos Tagging.

Come prima cosa, con "scikit-learn" è stato definito il calcolo delle metriche: *accuracy, F1. precision* e *recall*.

L'addestramento è stato effettuato su 3 epoche: di seguito, i risultati ottenuti nell'ultima epoca (*epoch* 2).

Metrica	*Risultato*
Accuracy	0,985
F1	0,984
Precision	0,985
Recall	0,985

4. Risultati esecuzione Test Set

Al termine della fase di *training* e *validation*, e generati i risultati, il modello è stato confrontato con un opportuno *benchmark* di riferimento.

Indicazioni in merito al Test Set sono state fornite all'interno del codice: prima della fase di *training*, il Test Set (così come avvenuto per il Dataset nella procedura di *data processing*) è stato adeguatamente lavorato, adattato e pulito.

Non è stato predisposto lo *shuffle* dei dati ed è stato creato un secondo file .json di salvataggio per i risultati ottenuti.

Di seguito, i risultati generati sul Test Set:

Metrica	*Risultato*
Accuracy	0,987
F1	0.987
Precision	0,987
Recall	0,987

5. Conclusione

Dai risultati ottenuti, è possibile assumere che il modello ha dimostrato ottime performance sia sul Dataset di addestramento che sul Test Set, con valori di *accuracy, precision, recall* e *F1-score* molto elevati e consistenti tra loro. In particolare:

- sul Dataset di *training*, l'*accuracy* e il *recall* raggiungono lo 0,985, con un *F1-score* di 0,984, indicando una capacità del modello di apprendere efficacemente i pattern presenti nei dati;
- sul Test Set, tutte le metriche si attestano a 0,987, confermando la buona generalizzazione del modello su dati non visti.

Questi risultati suggeriscono che il modello è in grado di svolgere il task proposto in modo preciso e bilanciato, senza un *overfitting* evidente.