Progetto 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.
L'obiettivo è stato duplice: da un lato, comprendere i meccanismi alla base di una pipeline NLP per la token *classification*; dall'altro, approfondire l'utilizzo dei modelli BERT in un compito di annotazione linguistica supervisionata.
Al termine del progetto, i risultati del codice sviluppato sono stati confrontati con un opportuno benchmark di riferimento.

1. Introduzione

Il presente progetto riguarda l'adattamento di uno script Python per la *Named Entity Recognition* (da qui NER) in *Part-of-Speech Tagging* (PoS Tagging). Entrambi i compiti appartengono alla più ampia categoria della "token classification" nel Natural Language Processing (NLP), ovvero processi in cui a ogni token di una sequenza viene assegnata una determinata etichetta.

La **NER** si occupa di indicare, in 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.

Sebbene entrambi i task condividano l'approccio del token-level classification, le etichette da predire e le dinamiche contestuali sono molto diverse: mentre nella NER i token da etichettare sono spesso sparsi e discontinui, nel POS tagging tutti i token richiedono un'etichetta. Questo comporta anche una diversa distribuzione delle classi, una diversa granularità semantica e una maggiore dipendenza dalla sintassi.

1.1 Directory di partenza

Il progetto è stato eseguito utilizzando l'infrastruttura di Google Colab, per poter sfruttare le risorse messe a disposizione: come un ambiente che consente una gestione flessibile di GPU (nello specifico la T4) e l librerie necessarie. Sono stati inoltre stati utilizzati modelli e pacchetti della libreria *Hugging Face Transformers*.

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 *italian-cased* di Bert non pre-addestrato per il Pos-Tagging.

1.2 Architettura adottata

Nell'impostare il progetto, come prima cosa è stato scelto il modello a cui appoggiarsi per il task di Pos-Tagging.

Per quanto riguarda il **modello**, in un primo momento si è optato per quello pre-addestrato "sachaarbonel/bert-italian-cased-finetuned-pos" (link di riferimento: https://huggingface.co/sachaarbonel/bert-italian-cased-finetuned-pos).

Le caratteristiche di questo modello permettevano di preservare la distinzione tra lettere minuscole e maiuscole, caratteristica fondamentale per la disambiguazione di nomi propri e acronimi, spesso utili anche in contesti

morfosintattici; inoltre, l'architettura è "cased" per l'italiano.

L'idea di partenza, quando si è pensato inizialmente di optare per questo modello, era di adattare le etichette Eagles a quelle UPOS utilizzate nel modello affinché le potesse riconoscere. A quel punto, passare il nostro Dataset al suo interno, eseguirne l'addestramento e, una volta generati dei risultati, fornirgli il Test Set per il confronto.

Vista la non adeguatezza del modello inizialmente adottato, si è in un secondo momento optato per un modello "foundational", non pre-addestrato per il task del Pos Tagging. Il modello in questione, adottato in modo definitivo per il progetto è: dbmdz/bert-base-italian-cased (link di riferimento: https://huggingface.co/dbmdz/bert-base-italian-cased).

In questo modo è stato possibile lavorare direttamente sul dataset Eagles fornito e addestrare BERT nell'esecuzione del task richiesto. Questo ha comportato:

- un tempo di training più lungo;
- una maggiore sensibilità alla qualità del dataset e alla corretta gestione delle etichette;
- la necessità di una definizione esplicita di ogni fase del pre-processing e del mapping.

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, in un primo momento, si è rivelato funzionale anche durante la conversione dei tag del dataset a quelli del modello in formato UPOS.

Inoltre, l'utilizzo dell'ambiente Colab ha richiesto l'installazione delle seguenti librerie:

- *transformers* per la gestione del modello e del tokenizer;
- *datasets* per la conversione e gestione dei dati in formati compatibili con la libreria *Hugging Face*;
- *scikit-learn* per il calcolo delle metriche;
- *evaluate* per valutazioni più modulari e compatibili con PoS tagging.

Infine, all'interno della riga di comando, sono state indicate informazioni necessarie per il *training* del modello che richiamano gli argomenti del codice. Uno di questi riguarda la *Learning Rate*, stabilita su $5e^{-5}$ affinché venissero aggiornati i pesi durante la fase di *training* con una velocità coerente. Successivamente è stata definita la funzione *LR Schedular* per regolarizzare la *Learning Rate* durante il processo di *training* e permettere al modello di convergere a un minimo.

2. Sviluppo del codice

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

2.1 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.

Successivamente, il dataset è stato suddiviso per il *training* (80% del totale) e *validation* (20% del totale). Per eseguire lo splitting del dataset, necessario per poter addestrare il nostro modello con la quantità proporzionata di dati, è stata quindi stabilita la variabile "split_index" per lo *slicing*.

Una volta diviso, il dataset è stato accuratamente tokenizzato nella variabile "tokenizer": si è usato l'Autotokenizer del modello scelto per il task, affinché si potesse associare uno specifico ID di riferimento a ogni token.

Nel corso del lavoro di tokenizzazione, si è presentato un problema a cui si è dovuto far fronte: il modello richiedeva determinate lunghezze di token specifiche e dunque si bloccava. Per tale motivo, è stato necessario indicare il *padding*, ovvero la lunghezza massima di riferimento per la tokenizzazione. Inoltre, è stato inserito all'interno della riga di comando un parametro relativo alla lunghezza dei token (stabilita su 128),

Attraverso il *mapping*, i token ottenuti sono stati associati ai tag di riferimento e, infine, i Dataframe sono stati convertiti in Dataset *Hugging Face* affinché il Dataset venisse visto e riconosciuto dal modello (preso anch'esso da Hugging Face): Pandas, infatti, non riconosce la funzione *.map* necessaria alla mappatura.

2.2 Problemi post training

Una volta trattato e adattato adeguatamente il Dataset all'interno del codice affinché fosse funzionale al Pos Tagging, si è potuto dare inizio alla fase di *training* del modello.

Questo ha portato ad affrontare due problematiche illustrate nei seguenti sottoparagrafi.

2.2.1 Etichette

Come già illustrato nel paragrafo 1.1, in un primo momento si è optato per un modello pre-addestrato per il Pos Tagging di Bert. Per questo motivo, a seguito della fase di *training* ci si è trovati davanti al problema di mancata assegnazione e riconoscimento delle etichette Eagles (dataset da noi utilizzato) da parte del modello.

L'errore si evinceva da metriche perfette già dalla *epoch* 0 (corrispondenti a 1.0 e senza variazioni nelle varie prove) e dalla sola presenza di *labels* "O" (corrispondenti a valore nullo) all'interno delle etichette predette e reali.

Per tale motivo, è stato necessario rendere "label_list" non più un set ma una lista ordinata di elementi e adattare le etichette del Dataset di riferimento alle etichette UPOS riconosciute dal modello selezionato per il progetto. Per farlo, è stato:

- realizzato un dizionario che convertisse un tag del dataset Eagles nel tag corrispondente UPOS;
- creata una funzione che li mappasse.

Questo passaggio ha implicato:

- la costruzione di un dizionario di mapping tra tag;
- la riduzione dell'informazione grammaticale ad una categoria principale;
- la perdita di alcune informazioni morfologiche (numero, persona, modo, tempo), che però non erano necessarie per il nostro task.

Successivamente, quando è stato compreso fosse più adeguato un modello non pre-addestrato per il Pos-Tagging, si è effettuato il training prima su etichette UPOS (come in un primo momento convertite) e successivamente Eagles per valutarne le differenze.

Per far sì che il modello leggesse adeguatamente le etichette Eagles, è stata realizzata una funzione "fix_labels" per evitare che PyArrow leggesse le etichette come stringhe e non come valori numerici, e per far sì che i valori nulli (NAN) venissero sostituiti con un *placeholder*.

Si è compreso che il formato UPOS non fosse quello più idoneo perché troppo generico rispetto all'etichettatura stabilita da Eagles, e per questo c'era un maggiore rischio di andare incontro a performance troppo ottimali.

Per i risultati ottenuti con le etichette UPOS ed Eagles, si rimanda al paragrafo XX

2.2.2. Metriche

Sempre per quanto riguarda il problema dei risultati insolitamente eccellenti, un secondo problema trattato ha riguardato le metriche utilizzate e i risultati di conseguenza ottenuti.

Si sono convertite le metriche inizialmente presenti nel codice, riferite al pacchetto "*seqeval"* (tipico per la NER), convertendole in metriche più adatte al Pos-Tagging grazie al pacchetto *evaluate*.

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

- **Accuracy**: proporzione di etichette corrette sul totale;
- **Precision**: accuratezza delle previsioni per ciascuna classe;
- **Recall**: copertura delle istanze reali per ogni classe;
- **F1-score**: media armonica di *precision* e *recall*.

2.3 Ulteriori dettagli e modifiche apportate

Dopo il lavoro di *Data Processing* (spiegato al paragrafo 2.1) 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*.

Per quanto riguarda la *Loss* di *default* questa è la *cross entropy loss:* questa è la più diffusa in letteratura, soprattutto per quanto riguarda i task di Pos-Tagging.

Infine, è stato stabilito un file *.json* in cui salvare i risultati ottenuti.

2 Risultati ottenuti

Di seguito, illustriamo i risultati ottenuti con l'etichettatura **UPOS**:

Epoca	***Accuracy***	***Precision***	***Recall***	***F1-score***
0	0.983	0.983	0.983	0.983
1	0.987	0.987	0.987	0.987
2	0.987	0,987	0.987	0.987

A seguire, invece, i risultati ottenuti con l'etichettatura **Eagles**:

Epoca	**Accuracy**	**Precision**	**Recall**	**F1-score**
0	0.983	0.983	0.983	0.983
1	0.986	0.986	0.986	0.986
2	0.987	0.987	0.987	0.987

Dai risultati ottenuti, nonostante la maggiore complessità di classificazione richiesta dal Dataset Eagles, questo è quello ha permesso di avere delle metriche più reali e meno generalizzate.

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 per quanto riguarda l'etichettatura scelta.

Indicazioni in merito al Test Set sono state fornite in diverse fasi 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.

A differenza di quanto eseguito per il Dataset in esecuzione, 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 con l'ultima epoca: si è deciso di valutare solo l'ultima epoca, in quanto rappresenta quella con i risultati più ottimali.

Metrica	*Risultato*
Accuracy	0.988
F1	0.988
Precision	0,988
Recall	0,988
Loss	0.045

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 le metriche prese in esame raggiungono lo 0.987 nell'ultima epoca, indicando una capacità del modello di apprendere efficacemente i pattern presenti nei dati;
- sul test set, tutte le metriche si attestano a 0.988, 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.

Il progetto ha dimostrato come sia possibile riutilizzare un'infrastruttura pensata per la NER e adattarla al POS Tagging, modificando opportunamente dataset, metriche, architettura e metodi di valutazione.