Università Politecnica Delle Marche

Dipartimento di Ingegneria dell'Informazione

LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Relazione: Tasks di NLP di articoli di giornale con BERT

Docenti:

Prof. Domenico Ursino

Dott. Michele Marchetti

Matricola: 1113404

Smargiassi Luigi

Anno Accademico 2023 / 2024

Indice

Sommario

Scopo di questa relazione è illustrare l'attività di fine tuning di due modelli derivanti da **BERT**. In particolare, un modello sarà addestrato al fine di effettuare la **classificazione** di testi derivanti da articoli di giornale, o estratti da notiziari, in classi che ne rappresentano il macro argomento principale. Il secondo, invece, sarà in grado di effettuare un task di **Named Entity Recognition**, al fine di estrarre una serie di parole chiave e topic dagli articoli, che potranno essere usati a loro volta per indicizzare una lista di articoli da consultare. Il vantaggio del secondo metodo è che può estrarre topic anche per la prima volta, ad esempio riconoscendo **pattern** dei nomi di persone o organizzazioni, tuttavia devono essere presenti nel corpo della notizia o nel titolo. Mentre il primo sarà in grado di ricavare una **categoria** a partire dall'intera sequenza di parole. Per entrambi i modelli saranno definite delle basi di teoria necessarie a comprendere il contesto, le caratteristiche dei dataset impiegati per il fine tuning, il **modello pre-trained** di partenza, le fasi di **pre-processing** dei dati e di attivazione del processo di training, ed, infine, sono discussi i **risultati**. In conclusione, è definita una possibile applicazione del modello.

Il codice relativo è reperibile in una repository Github.

Capitolo 1

Introduzione: NLP e BERT

Negli ultimi decenni, il **Natural Language Processing (NLP)** ha segnato una notevole evoluzione nell'ambito dell'informatica, con l'obiettivo di sviluppare sistemi in grado di comprendere e interagire con il linguaggio umano. Questo campo multidisciplinare, situato all'incrocio tra linguistica, intelligenza artificiale e scienze dei dati, ha conosciuto una rapida crescita grazie all'avanzamento delle tecnologie e all'impiego di modelli di linguaggio sempre più sofisticati. In questo contesto, emerge con particolare rilevanza l'introduzione di **BERT (Bidirectional Encoder Representations from Transformers)**, un modello di linguaggio che ha rivoluzionato il modo in cui affrontiamo le sfide legate alla comprensione automatica del testo. Nella seguente relazione, esploreremo il contesto evolutivo della NLP, analizzando come BERT si inserisce come una pietra miliare in questo percorso di sviluppo tecnologico.

1.1 NLP (Elaborazione del Linguaggio Naturale)

La Natural Language Processing è un campo vasto e complesso, il cui obiettivo principale è sviluppare la capacità di comprendere, interpretare e generare il linguaggio umano in tutte le sue sfumature con sistemi automatici. Sin dalle prime fasi, le sfide della NLP hanno riguardato la comprensione semantica, la traduzione automatica, l'analisi del sentiment, e molte altre applicazioni linguistiche. Con l'avanzare della tecnologia, l'adozione di tecniche basate su apprendimento automatico ha permesso di superare alcune delle limitazioni dei modelli più tradizionali, in particolare superando, nel corso degli anni, le prestazioni dei **modelli basati su regole** che in passato erano la principale soluzione. Tuttavia, la **comprensione del contesto** e delle **sfumature semantiche** rimaneva un nodo cruciale. È in questo scenario che BERT si presenta come un punto di svolta, introducendo l'idea innovativa di contestualizzare

le rappresentazioni delle parole in un testo bidirezionale, permettendo al modello di cogliere il significato delle parole in base al loro contesto specifico.

1.2 BERT (Bidirectional Encoder Representations from Transformers)

BERT, sviluppato da Google Research, è un modello di linguaggio basato sulla trasformazione di tipo transformer. L'elemento distintivo di BERT risiede nella sua capacità di elaborare contesti bidirezionali durante la fase di addestramento. In contrasto con modelli precedenti che consideravano solo il contesto unidirezionale, BERT è in grado di catturare le relazioni semantiche all'interno di una frase considerando sia le parole precedenti che quelle successive. Questo approccio ha rivoluzionato il modo in cui le rappresentazioni linguistiche sono apprese, consentendo a BERT di generare embedding contestualizzati che riflettono la complessità e la ricchezza del linguaggio umano. L'implicazione pratica di questa caratteristica è evidente nelle numerose applicazioni di successo, dalle traduzioni più accurate alla risoluzione avanzata di compiti complessi di NLP, come la risposta alle domande, l'analisi del sentiment e altro ancora.

In questa relazione, esamineremo più da vicino le caratteristiche salienti di BERT, il suo processo di addestramento, e le applicazioni che hanno segnato un significativo progresso nell'elaborazione del linguaggio naturale. Attraverso questa analisi dettagliata, cercheremo di fornire una visione chiara del ruolo centrale che BERT gioca nel panorama attuale della NLP e come le sue innovazioni stiano plasmando il futuro di questa disciplina.

Capitolo 2

Natural Language Processing (NLP)

La **Natural Language Processing (NLP)** costituisce un campo vasto e complesso, mirante a dotare i sistemi automatici della capacità di comprendere, interpretare e generare il linguaggio umano nelle sue varie sfumature. Fin dalle sue prime fasi, la NLP si è confrontata con sfide significative, tra cui la comprensione semantica, la traduzione automatica, l'analisi del sentiment e molte altre applicazioni linguistiche. Inizialmente, i modelli basati su regole dominavano il panorama, ma con l'avanzare della tecnologia, l'adozione di tecniche basate sull'apprendimento automatico ha segnato un cambiamento di paradigma. Nel corso degli anni, tali approcci hanno superato le limitazioni intrinseche dei modelli basati su regole, fornendo prestazioni più robuste e adattabili.

L'evoluzione della NLP è stata caratterizzata da una graduale transizione verso modelli più sofisticati e flessibili. Nonostante il successo iniziale degli approcci basati su apprendimento automatico, la **comprensione del contesto** e delle **sfumature semantiche** rimaneva una sfida critica. Modelli **rule driven**, basati su regole, pur avendo rappresentato una soluzione chiave in passato, mostravano limiti nella gestione della complessità del linguaggio naturale e nella cattura delle dinamiche contestuali. L'introduzione di tecniche basate su **Deep e Machine Learning**, in particolare **BERT** (Bidirectional Encoder Representations from Transformers) ha segnato un punto di svolta significativo in questo percorso evolutivo.

Con l'affermarsi della NLP basata su apprendimento automatico, sono emersi numerosi task specifici che richiedono una comprensione avanzata del linguaggio umano. Tra i task più significativi si annoverano la comprensione del linguaggio naturale, la traduzione automatica, l'analisi del sentiment, la risposta alle domande, e l'identificazione delle entità nominate. Questi compiti richiedono una comprensione profonda del contesto e delle relazioni semantiche presenti nei testi, rendendo essenziale l'utilizzo di modelli che siano in grado di cogliere la

complessità e la varietà del linguaggio [1].

I modelli basati su regole, che in passato rappresentavano la principale soluzione per alcuni di questi task, spesso facevano fatica a gestire la diversità linguistica e la complessità strutturale dei testi reali. L'avvento di modelli come BERT ha portato una svolta significativa, permettendo di ottenere risultati notevolmente superiori su una vasta gamma di compiti. Preallenato su grandi corpora di testo, BERT è in grado di catturare le informazioni contestuali necessarie per affrontare task specifici senza richiedere una rifinitura intensiva.

2.1 Applicazioni del NLP

Le piattaforme di posta elettronica, come Gmail, Outlook, ecc. utilizzano ampiamente l'NLP per fornire una serie di funzionalità del prodotto, come la **classificazione dello spam**, la **priorità** della posta in arrivo, l'**estrazione degli eventi** del calendario, il **completamento automatico**, ecc. Gli **assistenti vocali**, come Apple Siri, Google Assistant, Microsoft Cortana e Amazon Alexa, si affidano a una serie di tecniche NLP per interagire con l'utente, comprendere i suoi comandi e rispondere di conseguenza. Queste tecniche spaziano dai chatbot, ad aspetti di sistemi più strutturati, che comprendono **speech recognition e generation**.

I moderni **motori di ricerca**, come Google e Bing, che sono la pietra miliare dell'Internet di oggi, utilizzano pesantemente l'NLP per varie attività secondarie, come la comprensione delle query, l'espansione delle query, la risposta alle domande, il recupero delle informazioni, la classificazione e il raggruppamento dei risultati. I servizi di **traduzione automatica**, come DeepL, Google Translate, Bing Microsoft Translator e Amazon Translate, sono sempre più utilizzati nel mondo di oggi per risolvere un'ampia gamma di scenari e casi d'uso aziendali. Questi servizi sono applicazioni dirette della PNL.

La NLP è ampiamente utilizzata per risolvere diversi casi d'uso su **piattaforme di e-commerce** come Amazon. Si va dall'estrazione di informazioni rilevanti dalle descrizioni dei prodotti alla comprensione delle recensioni degli utenti. I progressi della NLP vengono applicati per risolvere casi d'uso in ambiti come la sanità, la finanza e la legge. Inoltre, la NLP costituisce la spina dorsale degli strumenti di correzione ortografica e grammaticale, come Grammarly e il controllo ortografico di Microsoft Word e Google Docs.

Jeopardy! è un popolare quiz show televisivo. Nello show, ai concorrenti vengono presentati indizi sotto forma di risposte e i concorrenti devono formulare le loro risposte sotto forma

di domande. IBM ha costruito l'intelligenza artificiale Watson per competere con i migliori giocatori del programma. Watson ha vinto il primo premio con un milione di dollari, più dei campioni del mondo.

La NLP, inoltre, viene utilizzata per costruire **grandi basi di conoscenza**, come il Knowledge Graph di Google, utili in una serie di applicazioni come la ricerca e la risposta alle domande. In particolare a questo processo si assegna anche il nome commerciale di wikification, infatti si utilizza per generare dei grafi connessi a partire da grandi documenti o siti web, ed estrarre informazione da database pre esistenti.

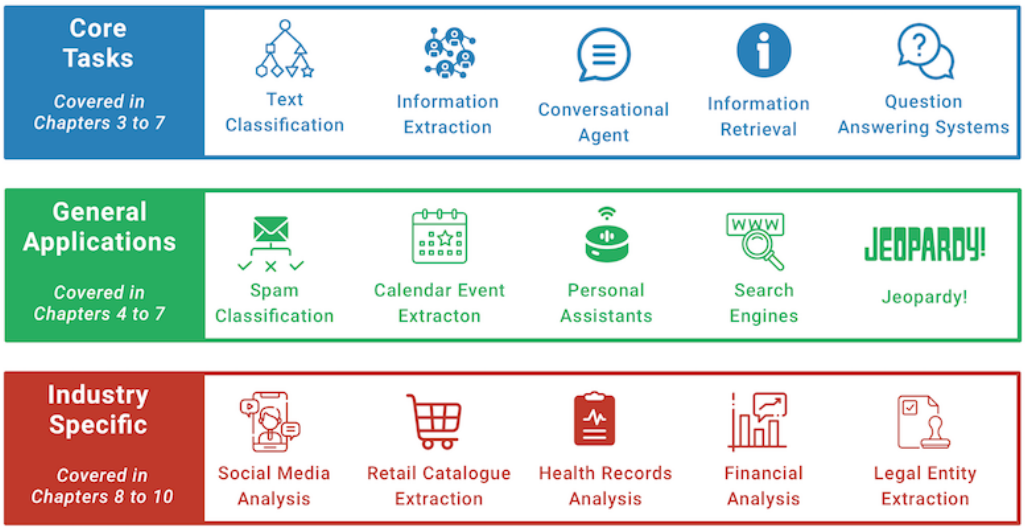


Figura 2.1: I task della NLP

Esiste un insieme di compiti fondamentali che compaiono frequentemente in vari progetti di NLP. A causa della loro natura ripetitiva e fondamentale, questi compiti sono stati ampiamente studiati. Se li conoscete bene, sarete pronti a costruire diverse applicazioni di NLP in tutti i settori verticali. Alcuni di questi compiti sono quelli riportati nella figura 2.1.

2.2 BERT

BERT ha introdotto un'innovativa architettura basata su transformer, permettendo al modello di contestualizzare le rappresentazioni delle parole in modo bidirezionale. Questa caratteristica distintiva ha consentito a BERT di superare le limitazioni dei modelli precedenti,

catturando le relazioni semantiche non solo tra le parole precedenti, ma anche tra quelle successive in una frase. L'approccio bidirezionale ha dimostrato di essere fondamentale per catturare la complessità e la ricchezza del linguaggio umano, consentendo a BERT di generare embedding contestualizzati che riflettono in modo accurato il significato delle parole in base al loro contesto specifico.

2.2.1 I Transformer

Nel 2017, l'articolo pubblicato dai ricercatori di Google, Attention is All you Need [2], apre una nuova pagina per il mondo delle reti neurali, che inevitabilmente cercava nuove vie per espandere le capacità dei sistemi, senza compromettere i costi delle applicazioni. Al contempo ULMFiT [3] portava le reti ricorrenti con memoria(LSMT) addestrate con tecniche di transfer learning a risultati pari allo stato dell'arte nel settore della classificazione di testo. Queste non sono che le pubblicazioni scientifiche di quelli che sono successivamente divenuti i due fenomeni commerciali principali del settore: GPT(Generative Pretrained Transformer) e BERT(Bidirectional Encoder Representation from Transformer), i quali progressi hanno superato di gran lunga quelli che erano i risultati dell'NLP precedenti.

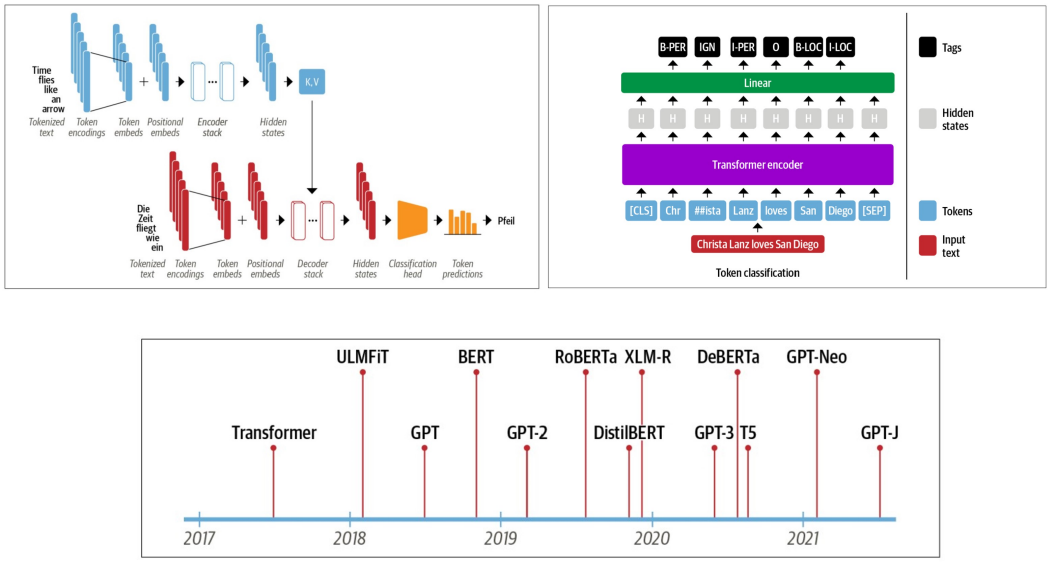


Figura 2.2: Struttura e timeline

L'importanza del contesto nei task di NLP è discussa nel capitolo precedente, questa porta le reti ricorrenti con memoria ad avere prestazioni superiori a quelle senza, in particolare in architetture *encoder-decoder* e *sequence to sequence* [4]. Queste hanno storicamente rilevanza

nei processi di mapping di parole in frasi tradotte in altre lingue, tuttavia subiscono drastici cali nelle prestazioni in presenza di input particolarmente ampi, poiché la memoria tende inevitabilmente a deteriorarsi o saturare. Una soluzione è il meccanismo dell'**attention**, creando uno stato intermedio, che poi è interamente utilizzato come input del layer della rete successivo, anche in questo caso la rete arriva a dei limiti legati alla dimensione dell'input del layer più profondo. Il passo logicamente successivo è il meccanismo della **self attention**. Essa consente ai modelli di apprendimento automatico di assegnare pesi dinamicamente a diverse parti di un input, enfatizzando le relazioni importanti e riducendo l'influenza di quelle meno rilevanti. In un modello di Transformer, consente al modello di considerare simultaneamente tutte le posizioni nei dati di input, sia passate che future, attribuendo pesi diversi a ciascuna posizione. Ciò si traduce in una maggiore capacità di catturare le relazioni a lungo termine e le dipendenze contestuali, rendendo il modello più adattabile a sequenze di dati complesse, come frasi o documenti. In NLP questo significa avere a disposizione il contesto continuamente, ed anzi di poter prescindere dall'ordine nel quale le parole compongono le frasi, il tutto senza dipendere dal meccanismo di back feeding, che rende particolamente onerose alcune RNN.

Capitolo 3

Classificazione

La classificazione rappresenta uno dei pilastri fondamentali nell'ambito del NLP. Questo capitolo esplorerà le sfide, le metodologie e le applicazioni della classificazione nell'ambito NLP. La **classificazione in NLP** è il processo di assegnare una o più **etichette predefinite** a un dato testo in base al suo contenuto. Le etichette possono rappresentare categorie, argomenti, sentimenti o qualsiasi altra informazione rilevante per il contesto specifico dell'applicazione. Prima dell'avvento dei modelli basati su trasformatori, gli approcci tradizionali alla classificazione NLP includevano l'uso di metodi **basati su regole**, **feature engineering** e modelli di apprendimento automatico più classici come le **SVM (Support Vector Machines)** e i modelli lineari. Questi metodi facevano affidamento su rappresentazioni manualmente progettate delle caratteristiche del testo.

L'emergere di modelli di linguaggio avanzati basati su **trasformatori**, come BERT, GPT ed altri, ha rivoluzionato la classificazione in NLP. Questi modelli apprendono rappresentazioni **contestualizzate** del linguaggio, catturando relazioni semantiche complesse in modo automatico. L'approccio pre-allenato e il fine-tuning per task specifici hanno migliorato significativamente le prestazioni nei compiti di classificazione. La **pre-elaborazione** del testo e la **tokenizzazione** sono passaggi critici nella classificazione NLP. La tokenizzazione suddivide il testo in unità più piccole, come parole o sottostringhe, rendendo più agevole l'elaborazione. La pre-elaborazione può includere la rimozione di stop words, la lemmatizzazione, che è processo che gestisce parole che si generano da inflessione di una stessa radice, e la gestione delle maiuscole/minuscole per standardizzare il testo.
La valutazione delle prestazioni di un modello di classificazione NLP è cruciale per comprendere la sua efficacia. Metriche come precision, recall, F1-score e l'accuracy vengono comunemente

utilizzate per misurare la qualità delle predizioni rispetto alle etichette di riferimento.
La classificazione NLP ha una vasta gamma di applicazioni pratiche, tra cui la categorizzazione di documenti, la sentiment analysis, la rilevazione di spam nelle email o di fake news, l'identificazione di temi in social media, e molto altro. L'utilizzo di modelli avanzati ha permesso di affrontare task complessi, consentendo un'applicazione più ampia e precisa della classificazione nell'ambito NLP.
La classificazione in NLP è una componente fondamentale per svariati task e applicazioni. L'evoluzione dei modelli di linguaggio, in particolare l'introduzione del **transfer learning**, ha segnato un cambiamento significativo, permettendo un'apprendimento automatico più profondo e contestualizzato.

3.1 Il dataset

Per sviluppare il modello che effettuerà classificazione del testo, si è impiegato il dataset **News Category Dataset**[5] [6], reperibile su Kaggle e sul portale Github del creatore.
Questo dataset contiene circa 210 mila titoli di notizie dal 2012 al 2022 provenienti dall'HuffPost. Si tratta di uno dei più grandi dataset di notizie,nasce con lo scopo di essere utilizzato come benchmark per una serie di compiti linguistici computazionali. L'HuffPost ha smesso di mantenere un ampio archivio di articoli di notizie qualche tempo dopo la prima raccolta di questo set di dati nel 2018, quindi non è possibile raccogliere un simile set di dati al giorno d'oggi. A causa dei cambiamenti nel sito web, ci sono circa 200 mila titoli tra il 2012 e il maggio 2018 e 10 mila titoli tra il maggio 2018 e il 2022. Per ciascun record nel dataset si hanno i seguenti attributi:

- **category**: la classe assegnata all'elemento, sono i valori che il modello andrà a stimare.
- **headline**: il titolo dell'articolo in questione.
- **authors**: lista degli autori che hanno contribuito all'articolo.
- **link**: URL dell'articolo sul sito originale.
- **short_description**: Breve sunto dell'articolo.
- **date**: data di pubblicazione dell'articolo.

Le categorie sono 42, in Listing 1, .Tuttavia, da un'analisi rapida, si evince che si tratta sia di categorie per indicizzare gli articoli in base all'argomento, ma anche di rubriche, come

```
Counter({
    'U.S. NEWS': 1377,        'COMEDY': 5400,            'PARENTING': 8791,
    'WORLD NEWS': 3299,       'CULTURE & ARTS': 1074,    'TECH': 2104,
    'SPORTS': 5077,           'ENTERTAINMENT': 17362,    'POLITICS': 35602,
    'WEIRD NEWS': 2777,       'ENVIRONMENT': 1444,       'EDUCATION': 1014,
    'CRIME': 3562,            'SCIENCE': 2206,           'WELLNESS': 17945,
    'BUSINESS': 5992,         'STYLE & BEAUTY': 9814,    'FOOD & DRINK': 6340,
    'MEDIA': 2944,            'QUEER VOICES': 6347,      'HOME & LIVING': 4320,
    'WOMEN': 3572,            'BLACK VOICES': 4583,      'TRAVEL': 9900,
    'MONEY': 1756,            'RELIGION': 2577,          'LATINO VOICES': 1130,
    'IMPACT': 3484,           'WEDDINGS': 3653,          'COLLEGE': 1144
    'PARENTS': 3955,          'ARTS & CULTURE': 1339     'STYLE': 2254,
    'GREEN': 2622,            'TASTE': 2096,             'HEALTHY LIVING': 6694,
    'THE WORLDPOST': 3664,    'GOOD NEWS': 1398,         'WORLDPOST': 2579,
    'FIFTY': 1401,            'ARTS': 1509,              'DIVORCE': 3426
})
```

Listing 1: Categorie originariamente presenti nel dataset e numero di elementi

ad esempio le categorie che terminano in "VOICES", oppure "FIFTY". Esse sono state sostituite in fase di pre processing con label quanto più simili fra quelle generali, nell'ottica di generalizzare il modello ad articoli provenienti da fonti quanto più svariate.

3.1.1 Accelerazione TPU su Kaggle

Prima di procedere con l'effettivo processo per l'addestramento del modello, in questa sezione discutiamo una tecnologia abilitante per i modelli più moderni. La **TPU**, o **Tensor Processing Unit**, si pone come un'ulteriore evoluzione del calcolo parallelo a livello hardware. Le TPU sono circuiti integrati specifici per le applicazioni (ASIC, Application Specific Integrated circuits) progettati da **Google** per accelerare i carichi di lavoro di apprendimento automatico. Le TPU sono progettate per eseguire rapidamente operazioni matriciali, il che le rende ideali per i carichi di lavoro di apprendimento automatico. È possibile eseguire carichi di lavoro di apprendimento automatico su TPU utilizzando framework come TensorFlow, Pytorch e JAX. L'accelerazione offerta da questa tecnologia ai processi di addestramento e inferenza dei modelli di AI è fondamentale, in modo particolare per casi molto complessi, come i LLM.

```
try:
    tpu_cluster = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Running on TPU', tpu_cluster.master())
except ValueError:
    tpu_cluster = None # It assigns None, indicating that no TPU is available

if tpu_cluster: # is not None
    tf.config.experimental_connect_to_cluster(tpu_cluster)
    tf.tpu.experimental.initialize_tpu_system(tpu_cluster)
    # use a distribution strategy related to the presence of TPU
    dist_strategy = tf.distribute.TPUStrategy(tpu_cluster)
else: # is None
    # use a distribution strategy related to the absence of TPU
    dist_strategy = tf.distribute.get_strategy()

# Number of Replica-s in distribution strategy
dist_strategy.num_replicas_in_sync
```

Listing 2: Attivazione accelerazione TPU

Per sopperire al costo di questi strumenti hardware, le soluzioni as-a-Service sono una risorsa preziosa.Cloud TPU è un servizio di Google Cloud che rende disponibili le TPU come risorsa scalabile, in modo particolare Kaggle offre come servizio gratuito, se richiesto, l'utilizzo di TPU a disposizione sui server. Se attivato dal codice del notebook, con il codice riportato in Listing 3, si definisce un oggetto relativo la distribution strategy da attivare in base alla disponibilità o meno delle TPU.

3.2 Preprocessing ed Addestramento

Come già riportato, la prima fase del preprocessing consiste nel sostituire alcune delle categorie con altre più generiche, raggruppandone alcune. A partire dalle 42 categorie originariamente presenti si sono ottenute 27 categorie, distributire come in figura 3.1.

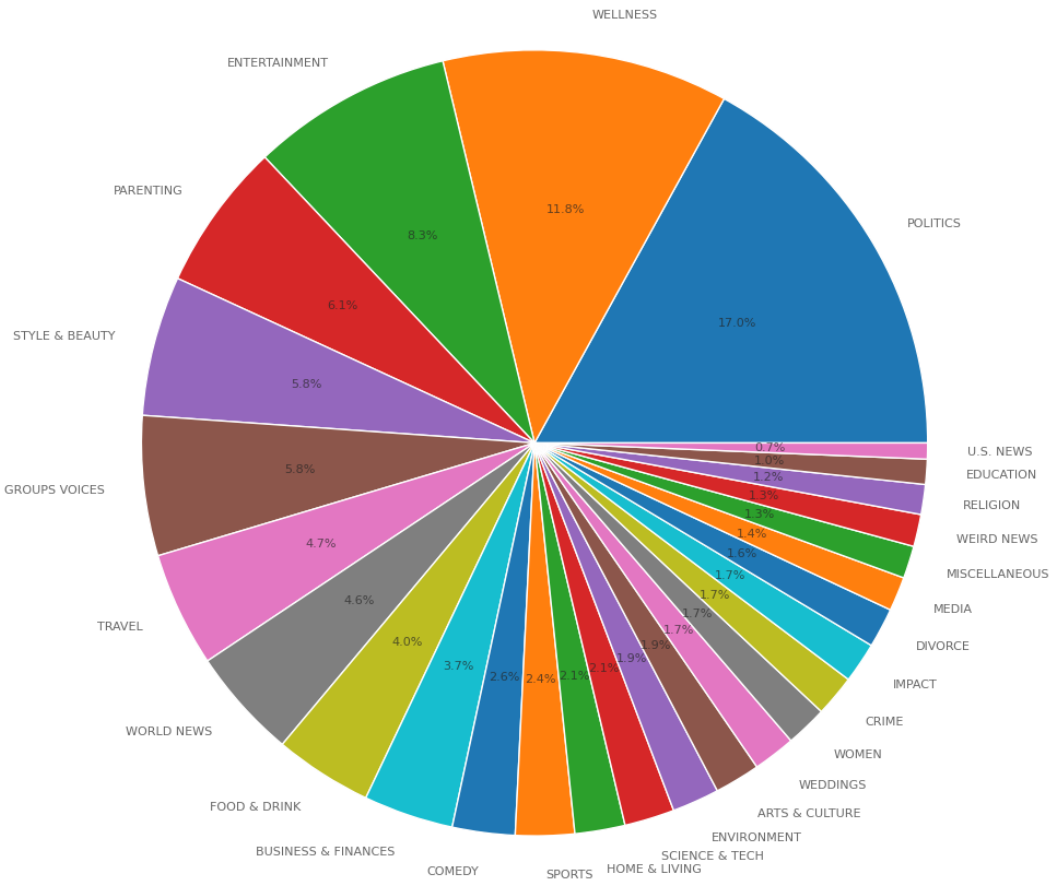


Figura 3.1: Diagramma a torta: distribuzione categorie

Si procede con l'eliminazione delle stopwords. Le stopwords sono parole che vengono comunemente filtrate durante il processo di analisi del testo in NLP (Elaborazione del Linguaggio Naturale) e information retrieval. Queste parole sono spesso rimosse poiché sono considerate di poco valore informativo o perché appaiono frequentemente in tutti i tipi di testi, non aggiungendo significato specifico al contenuto. Esse includono tipicamente parole comuni come articoli, preposizioni e congiunzioni. Oltre a consentire di concentrarsi sulle parole chiave e significative per l'interpretazione del testo, riduce al contempo la dimensionalità e migliorando l'efficienza computazionale. La selezione delle stopwords può variare a seconda del contesto e degli obiettivi specifici dell'analisi del testo, ma l'obiettivo principale è eliminare le parole che contribuiscono poco alla comprensione del contenuto informativo di un documento.
Per eseguire questa fase si è impiegato un file di stopwords per la lingua inglese, fornito da Kaggle. Queste sono rimosse dopo che sono state concatenati in un unico testo, titoli e corpi delle notizie. Su questo testo unico, si attiva é *word_ tokenize* di **nltk**. Questo restituisce una copia tokenizzata del testo, utilizzando il *tokenizer* di parole consigliato da NLTK (attualmente un *TreebankWordTokenizer* migliorato insieme a *PunktSentenceTokenizer* per la lingua specificata).
Infine, analizzando un token per volta, sono rimosse le stopwords e le parole vuote. In questa fase si attua tipicamente anche un **lemmatizer**. La lemmatizzazione è un processo di normalizzazione del testo in NLP che consiste nel ridurre le parole alle loro forme di base o lemmi. Un **lemma** è la forma di base di una parola, che rappresenta il suo significato principale.Si tratta di un ulteriore metodo di ridurre la dimensionalità del testo e ottenere una rappresentazione più compatta dei dati. Questo processo aiuta a consolidare diverse forme grammaticali di una parola in una forma standardizzata, facilitando la comprensione del contenuto del testo e riducendo la variabilità nei dati.

Infine, si passa al **fine tuning** del modello. Il modello utilizzato come partenza è **distilbert-base-uncased**, DistilBERT[7] è un modello di dimensioni ridotte, le prestazioni non sono molto inferiori al caso complesso. Infatti, è addestrato per **distillation** del modello BERT, il che significa che è stato addestrato per prevedere le stesse probabilità del modello più grande.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 133)]             0

 tf_distil_bert_model (TFDis  TFBaseModelOutput(last_h  66362880
 tilBertModel)               idden_state=(None, 133,
                             768),
                              hidden_states=None, att
                             entions=None)

 tf.__operators__.getitem (S  (None, 768)              0
 licingOpLambda)

 dropout_19 (Dropout)        (None, 768)               0

 dense (Dense)               (None, 27)                20763

=================================================================
Total params: 66,383,643
Trainable params: 66,383,643
Non-trainable params: 0
_________________________________________________________________
```

Listing 3: Report del fine-tuning

3.3 Risultati

In questa sezione, si commentano i risultati ottenuti dal fine tuning del modello. Per iniziare studiamo i valori che sono forniti come output del *fitting* del modello, essi sono memorizzati su una variabile, e consultabili con il metodo history. il grafico riportato in figura 3.2. Questo riporta i valori ottenuti calcolando la **loss function**, ovvero l'obiettivo, sia sui dati di *training*, che sul *validation set*.

Studiare l'andamento della loss e della validation loss durante il fine-tuning di modelli come BERT è una pratica comune e cruciale nell'addestramento di reti neurali. Esse forniscono una misura quantitativa delle prestazioni del modello durante il processo di fine-tuning. Una loss più bassa indica che il modello sta apprendendo bene dai dati di addestramento, tuttavia una tendenza ad adattarsi troppo ai dati di training può significare **overfitting**.

Quindi, monitorare la validation loss aiuta a identificare segnali di overfitting, e la perdita della capacità di generalizzare su nuovi dati.

L'andamento delle loss può guidare la decisione su quante **epoche** di fine-tuning sono necessarie. Un monitoraggio attento consente di evitare l'overfitting e di arrestare l'addestramento quando il modello ha raggiunto una buona generalizzazione. Nel caso specifico, dato che l'addestramento raggiunge un punto in cui non migliora più, al fine di evitare quanto descritto prima, alla dodicesima epoch il tuning è concluso.

Figura 3.2: Loss function di train e validation set per ciascuna epoch

Altro valore calcolato su entrambi i set, di training e di validation, è la **f1-score**, il grafico

in figura 3.3 ne mostra l'andamento al crescere delle epoch. Monitorare l'andamento della metrica F1 durante il fine-tuning è altrettanto importante. F1 è una metrica di valutazione delle prestazioni che tiene conto sia della **precision** che della **recall**. La sua combinazione offre una visione complessiva delle prestazioni del modello nel gestire **classificazioni positive** e **negative**, che in un contesto con numerose classi come quello in oggetto risulta essere compito non semplice. Un aumento dell'F1 indica un miglioramento complessivo della capacità del modello di discriminare correttamente tra le classi. Inoltre, in molti scenari di classificazione, le classi possono essere **sbilanciate**, con un numero significativamente maggiore di esempi di una classe rispetto a un'altra. F1 è particolarmente utile in questi contesti, in quanto tiene conto sia degli errori di falsi positivi che di falsi negativi, offrendo una visione equilibrata delle prestazioni. *Precision* e *recall*, da cui deriva F1, forniscono informazioni specifiche sugli **errori di tipo I** (falsi positivi) e **tipo II** (falsi negativi). Monitorare l'andamento di F1 durante il fine-tuning aiuta a garantire che il modello raggiunga un equilibrio tra precisione e recall, fornendo una valutazione completa delle sue capacità su diverse classi.

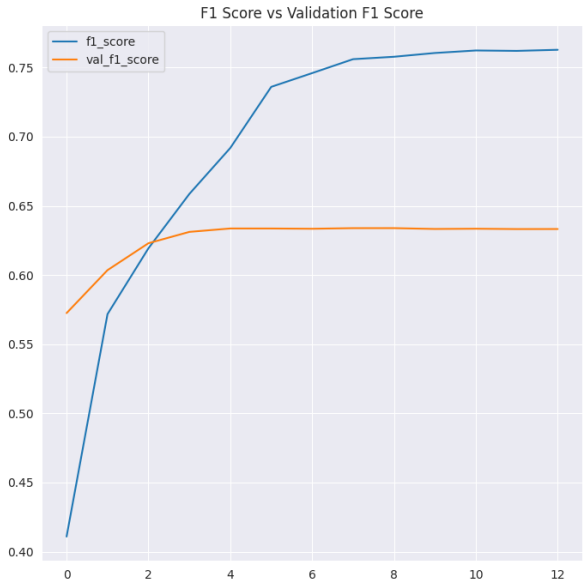


Figura 3.3: F1 score di train e validation set per ciascuna epoch

Infine, in figura 3.4, si riporta la **matrice di confusione** delle 27 categorie(classi) ottenuta a partire dall'applicazione del modello addestrato al **test set**. Come si può comprendere intuitivamente dalla colorazione delle celle, le stime sono per lo più corrette. I casi nei quali non si ha predominanza di predizioni corrette sono per le classi con meno articoli, in particolar modo per quelle molto varie, che spesso coincidono con i raggruppamenti di rubriche del quotidiano dal quale sono raccolti i dati, come 'GROUP VOICES', 'WEIRD NEWS' e

'MISCELLANEUS', che è la classe che è stata predetta con maggiori difficoltà.

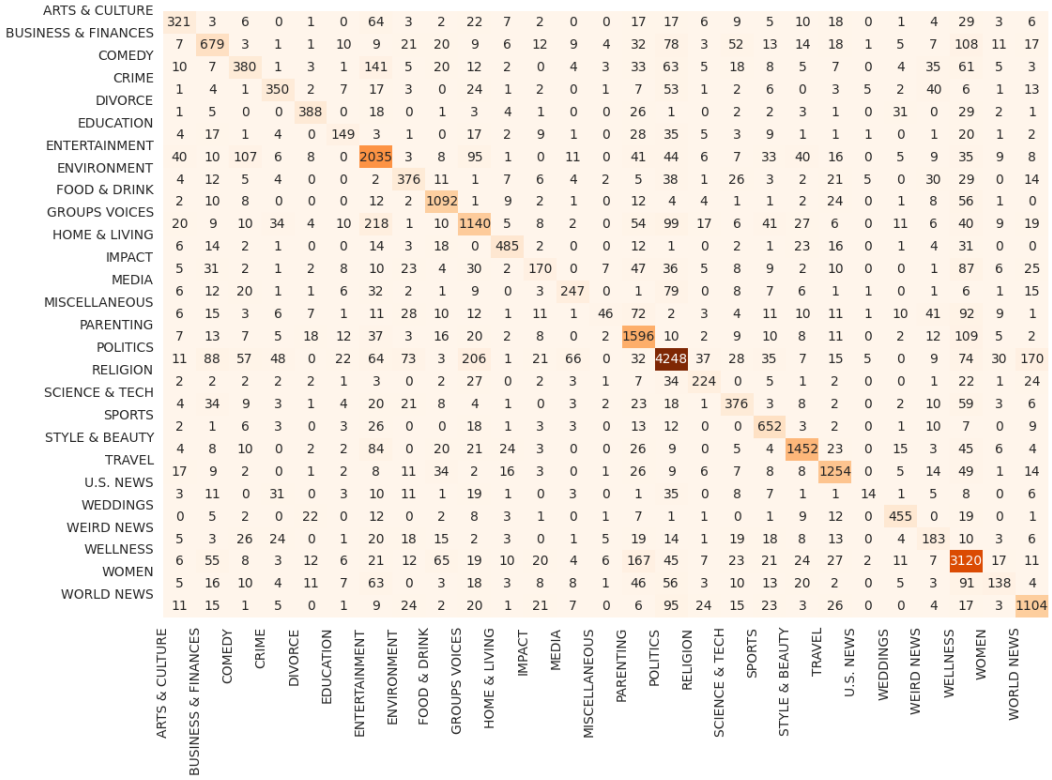


Figura 3.4: Confusion Matrix delle classi predette

Capitolo 4

Named Entity Recognition (NER)

Un settore particolarmente critico del NLP è l'estrazione di informazione, o **information extraction**, **IE**. L'estrazione di informazione per poi applicare eventualmente dei metodi di data mining per ricavare knowledge, o conoscenza, è fondamentale in varie applicazioni commerciali. I dati testuali sono una forma di dato non strutturato, quindi privo di schema, il che rende spesso impossibile o sconveninete applicare dei metodi automatici. L'estrazione della conoscenza, quindi, risulta essere un compito molto complesso, per la natura molto differente dei task che lo compongono, e per molti versi un metodo automatico assoluto risulta ancora fuori dalla portata. Tuttavia, stringendo il focus su particolari compiti è possibile costruire sistemi automatici che hanno prestazioni soddisfacenti.
In primo luogo, studieremo in questa relazione, solo dati testuali, anche se esitono molte soluzioni miste, ad esempio, che analizzano immagini e testo delle notizie presenti su notiziari online. I sistemi di IE hanno il compito di trovare e comprendere quanto contenuto in parti del testo complessivamente posto in input ad essi, presentandone, in output, una versione strutturata. Per farlo devono combinare conoscenze di linguaggio e dominio.
In questo ambito, si parla di Low Level IE, per quei task che studiano piccoli raggruppamenti di parole, che contengono informazione separata dal contesto. In particolare, il task che è descritto in questo capitolo è la **Named Entity Recognition**, o **NER**, si tratta di una parte del più grande insieme di task di IE. Essa consiste nel trovare e classificare nomi presenti nel testo, che possono comporsi di più token, questo svolge un ruolo molto importante alla base dei processi d'estrazione dei topic. Inoltre, viene utilizzato anche per indicizzare il contenuto di pagine web, per la ricerca per parole chiave.
In questo settore, alcuni modelli più semplici, come quelli basati su regole (automi a stati finiti per gruppi di parole) e alcuni derivati da Machine Learning hanno risultati migliori rispetto

altri task nei quali tali modelli hanno prestazioni scadenti. Tuttavia soffrono molto casi di **boundary error**, che possono sfuggire a metriche di valutazione classiche. Per questo le prestazioni di modelli basati su transfer learning sono superiori, in particolare, **BERT-NER**.

4.1 Il dataset

BERT-NER è un modello ottenuto dal fine tuning di BERT allo scopo di effettuare NER su testi generici, è addestrato sul dataset **CoNLL2003** [8]. Esso è stato rilasciato come parte dello shared task CoNLL-2003: **language-independent named entity recognition**. I dati sono costituiti da otto file che coprono due lingue: Inglese e Tedesco. Per ciascuna lingua sono presenti un file di addestramento, un file di sviluppo, un file di test e un file grande con dati non annotati. I dati in inglese sono stati presi dal Reuters Corpus. Questo corpus è costituito da notizie Reuters dell'agosto 1996 e dell'agosto 1997. Il testo per i dati tedeschi è stato preso dal corpus testuale multilingue dell'ECI. Questo corpus è composto da testi in molte lingue. La porzione di dati utilizzata per questo compito è stata estratta dal quotidiano tedesco Frankfurter Rundshau. Tutti e tre gli insiemi di addestramento, sviluppo e test sono stati ricavati da articoli scritti nel 1992.

CoNLLpp[9] è una versione corretta del dataset NER CoNLL2003, in cui le etichette del 5,38% delle frasi del set di test sono state corrette manualmente. Per completezza, sono stati inclusi il set di addestramento e il set di sviluppo di CoNLL2003. Il dataset è reperibile sul portale di Hugging Face.
Il dataset si compone di 3 set: training, test e validation, come si può vedere in Listing 4. Ciascun elemento ha 5 feature, una è l'id, i token, successivamente, ci sono una sequenza di label, che rappresentano il risultato atteso del modello, una serie di categorie. I tag sono espressi in formato IOB, che è sigla di *inside-outside-beginning*, che è tipico, i tag '0' sono associati a token sui quali non si riconosce un'entità, al contrario, se si riconosce un'entità, si segnala con un tag specifico l'inizio e tutti i successivi token associati allo stesso tag, segnalano che esso prosegue(inside). Il dataset è derivato da articoli di giornale, per cui ogni elemento può essere sia un titolo o parte di un articolo, ma anche il nome dell'autore o altre informazioni aggiuntive sull'articolo, come date o luoghi.

```
DatasetDict({
    train: Dataset({
        features: ['id', 'tokens', 'pos_tags', 'chunk_tags',
        'ner_tags', 'ner_tags_str'],
        num_rows: 14041
    })
    validation: Dataset({
        features: ['id', 'tokens', 'pos_tags', 'chunk_tags',
        'ner_tags', 'ner_tags_str'],
        num_rows: 3250
    })
    test: Dataset({
        features: ['id', 'tokens', 'pos_tags', 'chunk_tags',
        'ner_tags', 'ner_tags_str'],
        num_rows: 3453
    })
})
```

Listing 4: Test-Train Split del dataset

4.2 Preprocessing ed Addestramento

La fonte del dataset comporta un minor sforzo necessario ad adattarlo allo scopo di effettuare *fine tuning* di BERT. Infatti, il dataset è già originariamente diviso in training e test set, e il testo è già spezzato per non avere input di dimensioni eccessive, ed è presente il *padding* dove necessario. Dunque, alla fase di pre processing non resta il processo di pulizia del testo da stopword, esse sono ricavate da nltk [10]. Inoltre, sono stati eliminati tutti i caratteri esclusi lettere, sostituendo i simboli di interpunzione con spazi vuoti. Sono infine rimossi tag html e parti che coincidono con i pattern delle emoji in vari ambienti. Per concludere, il dataset attraversa una fase di *shuffling*.
I modelli linguistici (LLM), compresi quelli come BERT (Bidirectional Encoder Representations from Transformers), in genere utilizzano etichette numeriche per i token anziché stringhe per ragioni pratiche e di efficienza computazionale durante l'addestramento. Infatti, l'uso di etichette numeriche riduce l'impronta di memoria del modello durante l'addestramento. Le etichette numeriche sono più efficienti in termini di memoria rispetto alle etichette stringa, che richiedono uno spazio di archiviazione aggiuntivo. Poiché i modelli linguistici hanno spesso a che fare con grandi vocabolari e quantità massicce di testo, l'ottimizzazione dell'uso della memoria è fondamentale per la scalabilità. Inoltre, le tabelle di embedding lookup sono utilizzate per mappare i token (parole o sottoparole) in rappresentazioni vettoriali continue. Questi embeddings sono indicizzati da ID numerici. L'uso di etichette numeriche per i token semplifica il processo di ricerca degli embeddings, rendendolo più efficiente dal punto di vista computazionale. Infine, le etichette numeriche semplificano l'elaborazione in batch, consentendo calcoli più rapidi ed efficienti sui moderni acceleratori hardware (ad esempio, GPU o TPU). Le etichette stringa richiederebbero operazioni aggiuntive per la codifica e la decodifica, aggiungendo overhead computazionale. Motivi per cui si effettua una conversione ad etichette dei token numerici, per una rappresentazione standardizzata e coerente tra diversi framework e librerie e la possibilità di calcolare la loss function.

Il modello che si intende addestrare ha in output 3 valori, l'id dell'input, e le sequenze di valori dell'attention mask e di label assegnate, il tutto è salvato sulla variabile *tokenized_datasets*. Si procede quindi con il fine tuning di un modello già addestrato, questo avverrà attraverso la **classe Trainer** di *transformers*. Commentiamo, quindi, il codice riportato in Listing 5.

Per prima cosa, discutiamo il *model_checkpoint*, esso è una variante di BERT, simile al caso base, che tuttavia non lavora solo con testi **lower case**, infatti, questo è un aspetto spes-

```
model_checkpoint = "distilbert-base-cased"
model = AutoModelForTokenClassification.from_pretrained(
                                                    model_checkpoint,
                                                    id2label=id2label,
                                                    label2id=label2id)
data_collator = DataCollatorForTokenClassification(tokenizer=tokenizer)

args = TrainingArguments("distilbert-finetuned-ner",
                         evaluation_strategy = "epoch",
                         save_strategy="epoch",
                         learning_rate = 2e-5,
                         num_train_epochs=3,
                         weight_decay=0.01)
trainer = Trainer(model=model,
                  args=args,
                  train_dataset = tokenized_datasets['train'],
                  eval_dataset = tokenized_datasets['validation'],
                  data_collator=data_collator,
                  compute_metrics=compute_metrics,
                  tokenizer=AutoTokenizer.from_pretrained(model_checkpoint))

```

Listing 5: Codice Fine-Tuning

so critico in NER, la presenza di maiuscole nei nomi spesso ne determina l'associazione ad entità differenti. A partire da tale modello è stato addestrato anche BERT-NER. DistilBERT è, inoltre, una versione *light-weight* del modello principale di BERT. La scelta è ricaduta su questo per questioni di tempo necessario al fine tuning, date le dimensioni del dataset, e le lunghe code necessarie ad accedere agli acceleratori TPU VM v3.8 per i notebook Kaggle.
Per prima cosa si impongono al *Trainer*, una serie di parametri, fra cui il numero di **epoch** nel quale avverrà il fine tuning. A seguire, si costruisce la pipeline, utilizzando gli strumenti preposti a task di token classification, alla base del processo di NER: DataCollatorForTokenClassification e AutoModelForTokenClassification.
I **DataCollator** sono oggetti che formano un **batch** utilizzando come input un elenco di elementi del dataset. Questi elementi sono dello stesso tipo degli elementi di training set o evaluation set. Per poter costruire i batch, essi sono in grado di applicare alcune elaborazioni sul dato(come il padding). Alcuni di essi applicano anche un incremento casuale dei dati (come il mascheramento casuale, che è previsto anche da DistilBERT) al batch formato.
AutoModel, invece, è uno strumento per i programmatori, che permette di selezionare in maniera dinamica il modello migliore attualmente disponibile, in base alle specifiche fornite in input ad esso.

4.3 Token Classification

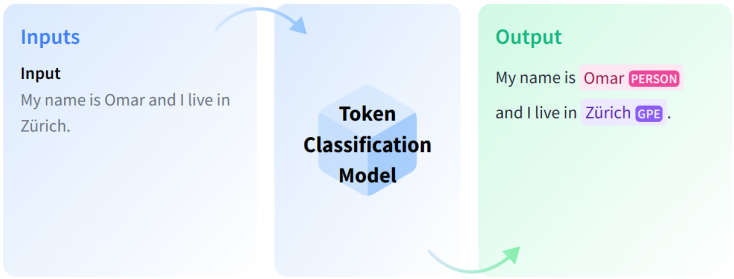
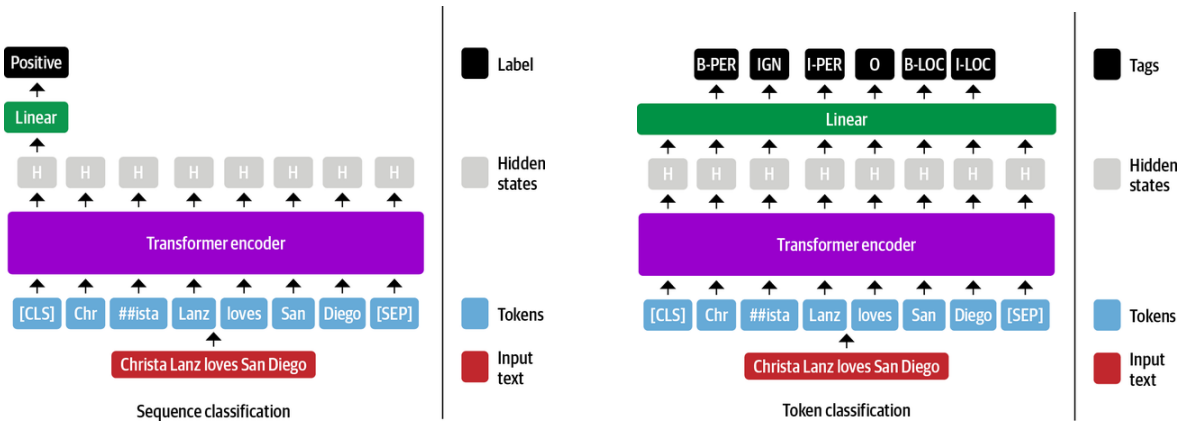


Figura 4.1: Esempio di token classification

La **token classification** è un task di NLP più generico della singola NER, che si può intendere come una delle sue possibili applicazioni al mondo del NLP[4]. Esso in NLP può essere inteso come task alternativo, o parallelo, alla **sequence classification**, che è utilizzata, ad esempio in ambito di *sentiment analysis*. La differenza fra i due processi sta nel modo nel quale l'hidden state alimenta il layer successivo: mentra il sequence classification le label sono generate da layer lineari, che prendono in input solo parte dello stato interno, in token classification tutto



(a) (b)

Figura 4.2: (a) Architettura Sequence Classification (b) Architettura Token Classification

lo stato interno alimenta il layer successivo, in maniera non dissimile a quanto avviene con l'input testuale in fase di tokenization, sfruttando appieno l'architettura di BERT. In Figura 4.2, le due architetture sono messe a confronto. In sostanza il problema di classificazione in caso di SC è dato da un elemento in input e in output N probabilità, legate alle N classi. Invece, in caso si TC, la dimensione dell'output cambia con la dimensione dell'input, come un problema di classificazione applicato a tutti gli elementi, che ha in uscita una serie di N probabilità per ciascun elemento.

In varie versioni di BERT, l'implementazione di questo task differisce in alcuni dettagli, come la gestinoe di token associati a subword, che in alcuni casi sono ignorate tutte meno la radice, in altri considerati tutti per avere corrispondenza 1:1 in output con il numero dei token.

4.4 Risultati

Per la stima della label da assegnare, si possono inserire delle metriche in fase di fine tuning, questo può portare a risultati anche molto diversi. La classe Trainer prevede che uno degli argomenti, *compute_metrics*, sia un metodo che restitutisce un dizionario con all'interno le metriche da impiegare per la selezione della label più adatta. Nel caso in questione si sono impiegate **precision**, **recall**, **f1-score** e **accuracy**. Questi valori contribuiscono al calcolo dello score finale con il quale si attesta la certezza con la quale si assegna la label.

In Listing 6, sono infine riportate le statistiche risultato del processo di fine tuning, proposte in output dal metodo *train*() del Trainer. Si ha che, per un modello relativamente semplice, rispetto agli standard della NLP, ed allenato per un numero di epoch basso, il risultato ottenuto è soddisfacente, con uno score della loss dell'8%.

```
TrainOutput(global_step=5268, training_loss=0.0810036782342191, metrics={
    'train_runtime': 12701.4994, 'train_samples_per_second': 3.316,
    'train_steps_per_second': 0.415, 'total_flos': 461068833627864.0,
    'train_loss': 0.0810036782342191, 'epoch': 3.0})
```

Listing 6: Output dell'esecuzione del Trainer

```
token_classifier = pipeline(
    "token-classification", model=checkpoint, aggregation_strategy="simple"
)

token_classifier("""
  Donald Trump has failed in an attempt to bring a case in the UK courts against
  the former MI6 officer who wrote a dossier linking him to Russia. He had been
  seeking to use data protection laws to sue Orbis Business Intelligence Ltd,
  the company run by Christopher Steele.
  ...
  A statement is expected from Mr Steele later today
  """)
```

Listing 7: Operazione di prova con il modello fine-tuned

Infine, in Listing 7, è riportato un esempio del funzionamento della pipeline realizzata, il modello prende in input un testo, derivante da una notizia di un quotidiano online, che presenta vari esempi di personalità pubbliche(nell'esempio, DOnald Trump, christofer Steele), luoghi(UK) ed organizzazioni(MI6). In output, riportato in Listing 8 si presenta la lista di gruppi di token ottenuti dal testo, associati all'indeice dell'input corrispondente, accompagnati dalla label assegnatagli e dallo score con cui questa è stata selezionata, .

```
[{'entity_group': 'PER',
  'score': 0.99716675,
  'word': 'Donald Trump',
  'start': 3,
  'end': 15},
 {'entity_group': 'LOC',
  'score': 0.99852425,
  'word': 'UK',
  'start': 64,
  'end': 66},
 {'entity_group': 'ORG',
  'score': 0.79941,
  'word': 'MI6',
  'start': 95,
  'end': 98},
 {'entity_group': 'LOC',
  'score': 0.9985837,
  'word': 'Russia',
  'start': 142,
  'end': 148},
 {'entity_group': 'ORG',
  'score': 0.9992711,
  'word': 'Orbis Business Intelligence Ltd',
  'start': 207,
  'end': 238},
...
 {'entity_group': 'PER',
  'score': 0.9380847,
  'word': 'Mr Steele',
  'start': 831,
  'end': 840}]
```

Listing 8: Output dell'operazione di prova con il modello fine-tuned

Capitolo 5

Sviluppi futuri

Una naturale prosecuzione di questo lavoro è quella di unire i due modelli in un unico LLM che esegue entrambi i task di classificazione e riconoscimento delle entità. Il **multitask learning** è un approccio per migliorare le prestazioni dei modelli di elaborazione del linguaggio naturale (NLP) su vari compiti. In questo contesto, si può migliorare ulteriormente le prestazioni addestrando i modelli a svolgere più compiti contemporaneamente. La complessità dell'addestramento di modelli di NLP per più task è particolarmente maggiore di quelli a singolo task. Inoltre, la dimensione dei dataset utilizzati per uno o l'altro addestramento, possono influire sui risultati finali, rendendo impossibile correggere il modello per non preferire l'accuratezza su un task agli altri. Si applicano quindi varie strategie assieme ad un **addestramento round robin**, una è quella del *Gradient Surgeon* [11].

L'idea quindi è di realizzare un modello che possa essere utilizzato dal backend di un'applicazione per consultare notizie per l'indicizzazione automatica di notizie da varie fonti. Eventualmente completando la pipeline con un frontend dotato di un sistema basato su RASA per intercettare delle richieste dell'utente e attivare una query sul database di notizie indicizzate, ricavata dagli interessi espressi dall'utente.

Elenco delle figure

Bibliografia

[1] S. Vajjala, B. Majumder, A. Gupta, and H. Surana. *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP Systems*. O'Reilly Media, 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[3] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

[4] L. Tunstall, L. von Werra, and T. Wolf. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, 2022.

[5] Rishabh Misra and Jigyasa Grover. *Sculpting Data for ML: The first act of Machine Learning*. 01 2021.

[6] Rishabh Misra. News category dataset. *arXiv preprint arXiv:2209.11429*, 2022.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[8] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[9] Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. Crossweigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5157–5166, 2019.

[10] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[11] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning, 2020.