Università Politecnica delle Marche
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione

SafeEdgeNER

Applicazione di tecniche di Named Entity Recognition a supporto della difesa

Micol Zazzarini
Andrea Fiorani
Antonio Antonini

Indice

Elenco degli Snippet di Codice

Elenco delle tabelle

Elenco delle figure

SafeEdgeNER

1 Introduzione

Il *Riconoscimento di Entità Nominate (NER, Named Entity Recognition)* è una tecnica di elaborazione del linguaggio naturale (NLP) volta a identificare e classificare automaticamente entità significative all'interno di un testo, come nomi di persone, organizzazioni, luoghi, date e altri tipi di entità definite. Questa metodologia si rivela particolarmente utile nel dominio della sicurezza, dove grandi volumi di dati testuali non strutturati devono essere analizzati rapidamente per estrarre informazioni critiche. Applicare la NER alla sicurezza permette di trasformare questi dati in informazioni strutturate, migliorando l'efficienza e l'efficacia delle operazioni analitiche. Il nostro progetto si propone di sviluppare un modello di NER ottimizzato per il contesto della sicurezza e della difesa, con un focus specifico sull'identificazione di entità di rilevanza strategica come piattaforme militari, armi, riferimenti temporali e documenti ufficiali. L'obiettivo principale è quello di fornire uno strumento che non solo automatizzi l'estrazione di queste informazioni, ma che lo faccia con un'elevata accuratezza e affidabilità, contribuendo a supportare decisioni critiche in scenari complessi. A tal fine, il progetto prevede la creazione e l'addestramento di modelli avanzati basati su tecniche di deep learning, come le reti neurali trasformative, per garantire prestazioni di alto livello anche su testi di dominio specifico e linguisticamente complessi.
Le applicazioni di una NER ottimizzata per la sicurezza sono numerose e promettenti. In ambito di intelligence, ad esempio, un sistema NER avanzato può essere utilizzato per analizzare documenti classificati, articoli di notizie, rapporti operativi o comunicazioni intercettate, consentendo di individuare rapidamente minacce emergenti o schemi di comportamento. Nel contesto della gestione delle crisi, la NER può facilitare il monitoraggio in tempo reale di eventi critici, identificando luoghi, attori coinvolti e cronologie rilevanti. In ambito forense e investigativo, invece, l'automazione dell'estrazione di informazioni può accelerare l'analisi delle prove e contribuire alla costruzione di casi più solidi.
Oltre alle applicazioni dirette alla sicurezza, questa tecnologia può essere integrata con altri sistemi di elaborazione linguistica, come il *Topic Modeling* o la *Sentiment Analysis*, per creare soluzioni analitiche multidimensionali. Ad esempio, combinando una NER con tecniche di clustering e visualizzazione, è possibile costruire dashboard interattive che consentano agli analisti di navigare attraverso grandi set di dati in modo rapido e intuitivo. Infine, l'adattabilità dei modelli NER a diverse lingue e domini apre la strada a un impiego internazionale, contribuendo alla cooperazione tra organizzazioni di sicurezza e intelligence su scala globale.

2 Il Dataset e il Formato CONLL 2003 IOB2

Il dataset utilizzato è basato sul *corpus re3d* ed è stato adattato per migliorare la qualità dei dati e l'applicabilità al nostro task. Contiene documenti relativi al conflitto in Siria e Iraq, con annotazioni che includono entità come "Location", "MilitaryPlatform", "Weapon", "Temporal", e altre categorie pertinenti. Sono state apportate modifiche significative, tra cui la rimozione di etichette con bassa affidabilità e la gestione di entità sovrapposte mantenendo quelle con lo span maggiore. Il dataset è stato convertito nel formato *CONLL 2003* utilizzando la codifica *IOB2*, in cui ogni token è etichettato come "B" (beginning) o "I" (inside) per indicare l'inizio o la continuazione di un'entità, oppure "O" per i token non facenti parte di alcuna entità. Questo formato facilita l'integrazione con strumenti standard di NLP e garantisce coerenza nelle annotazioni, rendendo il dataset una risorsa preziosa per addestrare e valutare modelli di NER in ambito sicurezza. Nella Tabella 1 sono presenti maggiori

informazioni sulle entità del nostro dataset.

Tabella 1: Descrizione delle entità presenti nel dataset

Entità	**Descrizione**
DocumentReference	Un identificatore unico (o semi unico) per un documento.
Location	Un punto, un'area o una caratteristica specifica sulla Terra. Può essere nominata (es. nome di una città) o riferita tramite coordinate.
MilitaryPlatform	Una nave, un aereo, un veicolo terrestre o un sistema militare su cui possono essere montate armi. Può essere indicata con la classe astratta (es. "*carro armato*") o una designazione specifica (es. "*T-14 Armata*").
Money	Una quantità di denaro o riferimento a una valuta.
Nationality	Descrizione di un'identità nazionale, religiosa o etnica.
Organisation	Un gruppo di persone, un governo, un'azienda o una famiglia. Può essere nominata (es. "*L'esercito britannico*") o contestuale (es. "*L'esercito*"). Può anche essere indicata tramite una classe astratta (es. "*ribelli*").
Person	Una persona specifica. Può essere indicata tramite un nome (es. "*Barack Obama*"), un titolo (es. "*Presidente degli Stati Uniti*") o una combinazione di entrambi (es. "Presidente Obama").
Quantity	Una quantità o misura di qualcosa (es. "*1 kg*").
Temporal	Una data, un'ora o un intervallo temporale specifico (es. "*10:30 PST*", "*12/09/2016*", "*La prossima settimana*").
Weapon	Un'arma. Può essere indicata con una classe astratta (es. "*fucile*") o con un nome specifico (es. "*AK-47*").

2.1 Exploratory Data Analysis (EDA)

Per comprendere a fondo il dataset e impostare correttamente il progetto, abbiamo condotto un'analisi esplorativa dei dati (EDA), fondamentale per identificare le caratteristiche principali del corpus e le eventuali problematiche. Questa fase preliminare è essenziale per ottenere una panoramica completa delle variabili coinvolte, della distribuzione delle entità e delle etichette, e per individuare possibili anomalie o lacune nei dati che potrebbero influenzare negativamente il processo di addestramento del modello.
Il primo passo è stato il caricamento del dataset, originariamente fornito in formato CONLL-03, e la sua conversione in un formato tabellare mediante l'utilizzo di *Pandas*, per facilitarne la manipolazione e l'analisi. Una rapida visualizzazione delle prime righe del dataset ha permesso di esaminare la struttura dei dati, confermando la presenza di token, etichette IOB (*Inside-Outside-Beginning*), e annotazioni associate alle entità. Da qui, abbiamo iniziato un'analisi descrittiva dei dati, calcolando statistiche di base, come il numero totale di righe, il numero di parole uniche e la distribuzione delle entità. Come

Statistiche	**Dataset di Addestramento**	**Dataset di Test**
Word Count	20030	5258
Unique Words	3752	1738
Top Word	the	the
Frequency of Top Word	959	266
IOB Label Count	3	3
Entity Count	11	11

Tabella 2: Statistiche di base del dataset di addestramento e di test

mostrato nella Tabella 2, il dataset di addestramento comprende 20.030 token, significativamente superiore ai 5.258 presenti nel dataset di test. Questo squilibrio riflette una normale distribuzione dei dati, in cui la maggior parte delle osservazioni viene utilizzata per addestrare il modello. Tuttavia, l'analisi evidenzia una varietà di parole uniche relativamente limitata (3.752 nel set di addestramento e 1.738 nel set di test), un'indicazione che i token si ripetono frequentemente. Il token più comune, "the", appare con una frequenza elevata in entrambi i dataset, un fenomeno atteso data la natura

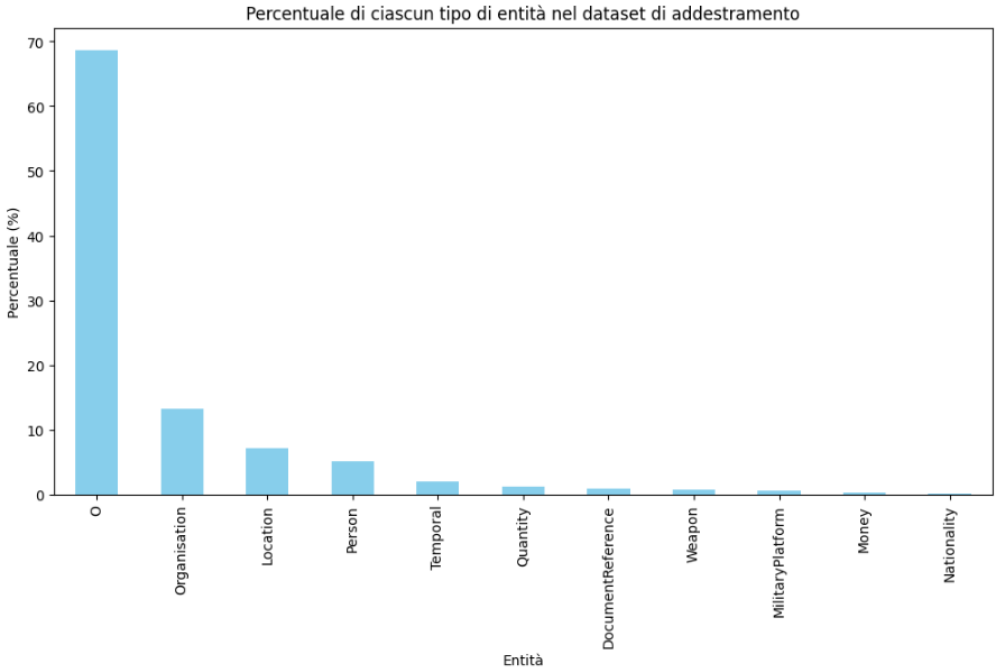


Figura 1: Percentuale di entità per tipo nel Training Set

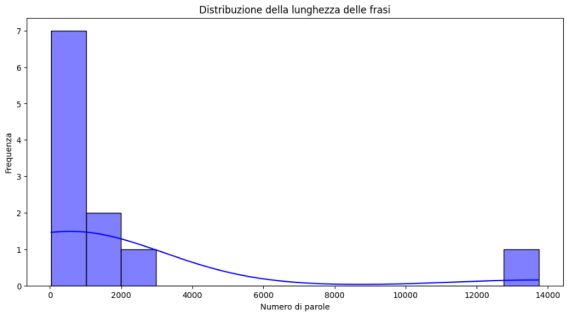
predominante delle parole funzionali in testi di dominio generico.
Un altro aspetto cruciale è stato l'analisi delle entità annotate e della loro distribuzione. Come riportato nella Tabella 3, il tipo di entità più comune è "O", che rappresenta i token non annotati. Tuttavia, tra le entità nominate, "Organisation", "Location" e "Person" emergono come le categorie dominanti, con frequenze significativamente più alte rispetto ad altre, come "MilitaryPlatform" o "Money". Questo sbilanciamento indica una maggiore disponibilità di dati per le categorie principali, ma anche la necessità di un'attenzione particolare per le entità con supporto limitato durante l'addestramento, in quanto potrebbero essere più difficili da riconoscere con precisione. L'analisi delle percentuali relative

Entità	**Frequenza (Training Set)**	**Frequenza (Test Set)**
O	13747	3672
Organisation	2643	695
Location	1428	346
Person	1016	256
Temporal	402	101
Quantity	244	62
DocumentReference	196	28
Weapon	151	31
MilitaryPlatform	111	42
Money	56	13
Nationality	36	12

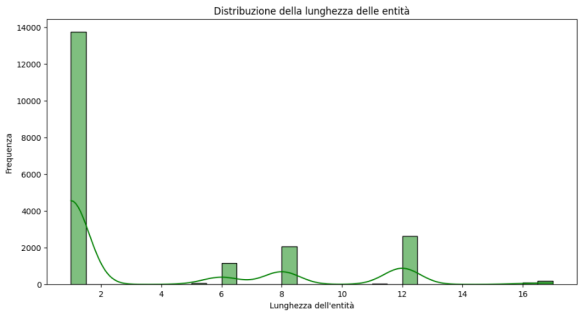
Tabella 3: Entità uniche nei dataset di addestramento e di test

alle entità per tipo, rappresentata nella Figura 1, mostra come le entità principali, come "Organisation" e "Location", coprano la maggior parte del set di dati. Questo suggerisce che il modello potrebbe imparare a generalizzare meglio per queste entità rispetto a categorie meno rappresentate. Tuttavia, una distribuzione così sbilanciata richiede un'attenta gestione, ad esempio tramite tecniche di bilanciamento dei dati o metriche di valutazione specifiche per classi con basso supporto, come vedremo in seguito nella fase di ottimizzazione dei risultati. Un aspetto particolarmente interessante riguarda la lunghezza delle frasi e delle entità nel dataset. La Figura 2a evidenzia che la maggior parte delle frasi nel set di addestramento è breve. Questo potrebbe riflettere la struttura dei documenti originali, suggerendo che molte frasi siano segmentate in unità concise. Parallelamente, l'analisi della lunghezza

delle entità, mostrata nella Figura 2b, rivela che la maggior parte delle entità è composta da uno o due token, con una distribuzione più rarefatta per entità più lunghe. Questo dato è particolarmente rilevante, poiché le entità brevi tendono a essere più semplici da riconoscere, mentre quelle più lunghe possono presentare sfide maggiori, soprattutto se contengono termini rari o non canonici. Abbiamo



(a) Distribuzione della lunghezza delle frasi nel Training Set.



(b) Distribuzione della lunghezza delle entità nel Training Set.

infine esaminato la distribuzione delle etichette IOB (Figura 3), confermando una netta predominanza di token con etichetta "O". Questa osservazione sottolinea la necessità di gestire uno squilibrio nelle etichette, poiché la predominanza di token non annotati potrebbe portare il modello a sottovalutare la rilevanza delle entità. È fondamentale adottare tecniche per garantire che le entità annotate abbiano un impatto sufficiente durante il processo di ottimizzazione, ad esempio utilizzando pesi personalizzati nelle funzioni di perdita.

Questa fase di EDA ci ha fornito una comprensione dettagliata del dataset e delle sue caratteristiche

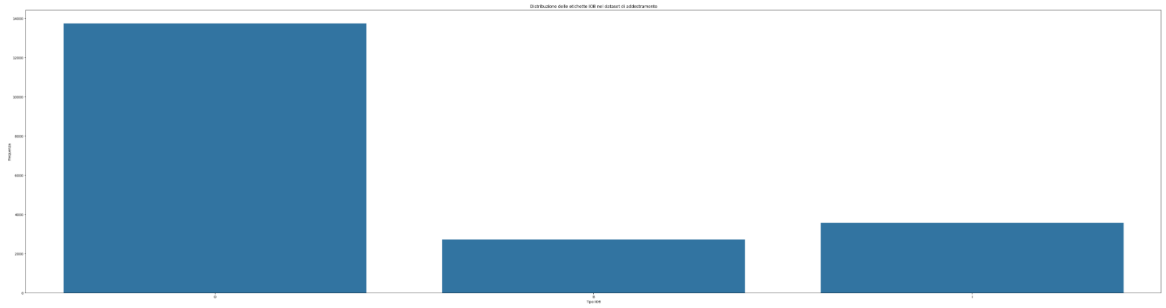


Figura 3: Distribuzione etichette IOB

principali. Abbiamo identificato punti di forza, come la ricchezza delle annotazioni per alcune categorie, e punti critici, come lo sbilanciamento tra entità, la scarsità di supporto per alcune categorie e la predominanza di frasi brevi. Queste osservazioni guideranno il processo di addestramento e ottimizzazione del modello, assicurandoci di affrontare adeguatamente le sfide emerse durante questa fase preliminare.

3 Il nostro progetto

Nel presente lavoro abbiamo affrontato il task di Named Entity Recognition (NER) utilizzando un approccio a più fasi, impiegando tre modelli di apprendimento automatico di crescente complessità. Inizialmente, abbiamo utilizzato un modello basato su *Conditional Random Fields (CRF)*, un metodo classico per l'analisi di sequenze che si è rivelato efficace nel riconoscere le entità nominali in un contesto di apprendimento supervisionato. Tuttavia, data l'evoluzione dei modelli di linguaggio, siamo poi passati a un approccio più avanzato, utilizzando un modello pre-addestrato di *BERT (Bidirectional Encoder Representations from Transformers)* per la classificazione di sequenze. BERT ha migliorato significativamente le performance grazie alla sua capacità di catturare la semantica del contesto bidirezionale, sfruttando i vasti dati di addestramento su cui è stato pre-addestrato. Successivamente, abbiamo implementato un modello ancora più sofisticato, *RoBERTa*, che rappresenta una versione ottimizzata di BERT. RoBERTa, infatti, si distingue per un miglioramento nell'efficienza dell'addestramento e una maggiore robustezza nelle performance, grazie a un addestramento su un numero maggiore di dati e un'architettura perfezionata.
Sebbene questi approcci abbiano portato a dei minimi miglioramenti nelle prestazioni, il principale ostacolo da affrontare è stato lo sbilanciamento del dataset, che ha influenzato negativamente l'efficacia dei modelli, rendendo la corretta identificazione di entità meno frequenti particolarmente difficile. Abbiamo perciò applicato diverse tecniche di ottimizzazione, ad esempio la gestione delle classi con pesi differenziati e l'uso di tecniche di oversampling per bilanciare la distribuzione delle entità nel dataset. Grazie a queste ottimizzazioni, combinate con l'aumento della complessità dei modelli, siamo riusciti a ottenere progressivi miglioramenti nei risultati, fino a raggiungere una performance soddisfacente.
Questo lavoro ha dunque mostrato come l'applicazione di modelli avanzati di linguaggio e l'adozione di tecniche di ottimizzazione possano portare a significativi miglioramenti nelle performance di un task di NER, dimostrando l'efficacia dei modelli basati su Transformers come BERT e RoBERTa in combinazione con approcci mirati per affrontare problematiche legate alla qualità e distribuzione dei dati.

3.1 Conditional Random Fields (CRF)

Il CRF è un modello probabilistico discriminativo utilizzato per sequence labeling. Modella la probabilità condizionale di una sequenza di etichette Y dato un insieme di feature X:

$$P(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_t, y_{t-1}, X, t)\right) \tag{1}$$

dove f_k rappresenta le feature del modello, λ_k i relativi pesi, e $Z(X)$ è una funzione di normalizzazione per garantire che $P(Y|X)$ sia una distribuzione valida.
Il CRF è particolarmente utile nel NER perché considera le dipendenze tra etichette consecutive, garantendo che le previsioni siano coerenti. E' dunque vantaggioso in quanto modella dipendenze sequenziali complesse, non fa assunzioni di indipendenza tra le feature, e produce previsioni coerenti e globalmente ottimali per l'intera sequenza. Presenta tuttavia delle limitazioni. Può essere infatti computazionalmente intensivo per dataset molto grandi, e richiede una progettazione accurata delle feature per ottenere buone prestazioni, come mostrato in seguito.

Pipeline

La pipeline si sviluppa nelle seguenti fasi principali: il **caricamento dei dati**, l'**estrazione delle caratteristiche** dalle frasi, l'**addestramento del modello CRF**, la **valutazione delle prestazioni** e la **visualizzazione dei risultati** tramite la matrice di confusione e il Classification Report.
Come già anticipato, i dati di addestramento e test sono forniti in formato *CoNLL 2003 (IOB2)*. I files contengono una sequenza di token con le rispettive etichette, e ogni frase è separata da una riga vuota. Attraverso una funzione di loading abbiamo dunque caricato i dati e li abbiamo organizzati in liste di parole e tag.
A seguire, abbiamo implementato l'estrazione di caratteristiche. Si tratta di una fase cruciale in una pipeline di apprendimento automatico, in cui si trasformano i dati grezzi in un insieme strutturato di

informazioni (feature) che il modello può utilizzare per fare previsioni. Nel contesto del task di Named Entity Recognition, questa serve a rappresentare ogni token di un testo attraverso proprietà rilevanti che ne catturano l'identità lessicale, il contesto sintattico, e la posizione nella frase. Le caratteristiche per ogni token sono estratte utilizzando la funzione `sent2feats` (1). Tra le caratteristiche incluse vi sono:

- Token corrente e contesto (parole precedenti e successive).
- POS-tagging per i token precedenti, correnti e successivi.

Questo è un approccio tradizionale, tipico di modelli come il CRF, dove è necessario costruire esplicitamente un set di caratteristiche per ogni token (ad esempio, etichetta di parte del discorso, contesto di parole, ecc.). Risulta dunque utile per:

- Catturare il contesto locale: Le feature includono parole precedenti e successive rispetto al token corrente, permettendo al modello di comprendere il significato in relazione alle parole circostanti.
- Utilizzare informazioni sintattiche: L'uso di tag POS (part-of-speech) fornisce indicazioni sul ruolo grammaticale del token, come se è un verbo, un nome, o un aggettivo.
- Gestire i confini della frase: Inserendo marcatori come $< S >$ e $< /S >$ per i confini iniziali e finali della frase, il modello riconosce dove inizia e finisce il contesto utile.

Listing 1: Estrazione delle caratteristiche

```
def sent2feats(sentence):
    feats = []
    sen_tags = pos_tag(sentence)
    for i in range(len(sentence)):
        word_feats = {'word': sentence[i]}
        if i == 0:
            word_feats['prevWord'] = '<S>'
        elif i == 1:
            word_feats["prevWord"] = sentence[0]
            word_feats["prevSecondWord"] = "</S>"
        else:
            word_feats["prevWord"] = sentence[i - 1]
            word_feats["prevSecondWord"] = sentence[i - 2]
        if i == len(sentence) - 2:
            word_feats["nextWord"] = sentence[i + 1]
        # ... altre caratteristiche qui ...
        feats.append(word_feats)
    return feats
```

Una volta estratte le caratteristiche, il passo successivo è addestrare il modello CRF utilizzando la libreria `sklearn_crfsuite`. La funzione `train_seq` (2) esegue l'addestramento, testando successivamente il modello sui dati di sviluppo (*validation set*). Viene utilizzato l'algoritmo `LBFGS` per l'ottimizzazione del modello.

Listing 2: Addestramento del modello CRF

```
def train_seq(X_train, Y_train, X_dev, Y_dev):
    crf = CRF(algorithm='lbfgs', c1=0.1, c2=10, max_iterations=50)
    crf.fit(X_train, Y_train)
    labels = list(crf.classes_)

    y_pred = crf.predict(X_dev)
    print("F1-score (weighted): ", metrics.flat_f1_score(Y_dev, y_pred, average='
    ↪ weighted', labels=labels))
```

Il modello viene addestrato con il set di addestramento `X_train` e `Y_train` (caratteristiche e etichette), e successivamente viene testato sui dati di sviluppo `X_dev` e `Y_dev`. Vengono calcolati i punteggi di F1 e il report di classificazione, che forniscono una panoramica delle prestazioni del modello su ciascuna

classe di entità. Per una miglior visualizzazione delle performance del modello è stata inoltre utilizzata la matrice di confusione utilizzando *Seaborn*, per rappresentare visivamente le predizioni corrette e gli errori.

Risultati

I risultati ottenuti dal modello CRF sono riportati di seguito nel Classification Report in tabella 4 e nella matrice di confusione in figura 4.
L'F1-score weighted è pari a 0.773, indicando che il modello ha buone prestazioni globali. Tuttavia, le prestazioni variano significativamente tra le diverse classi. La precisione e la recall mostrano una forte disparità tra le entità comuni (ad esempio, "O" per parole non classificate) e quelle rare o meno rappresentate. La classe "O" (non entità) ha la precisione e la recall più elevati (precisione 0.826, recall 0.973), grazie all'abbondanza di esempi e alla semplicità del task. Classi come "B-Organisation", "I-Organisation", "B-Person" e "I-Person" mostrano performance mediamente buone con F1-scores che vanno da circa 0.54 a 0.65. Le classi problematiche sono quelle meno rappresentate. Alcune classi,come "B-Nationality", "I-DocumentReference", "B-Money", "B-Weapon", hanno precisione e recall nulli (F1-score 0.0). Questo è dovuto alla scarsità di esempi nel dataset, che rende difficile al modello apprendere le caratteristiche di queste classi. Per quanto riguarda "B-Temporal" e "I-Temporal", sebbene abbiano un F1-score moderato (rispettivamente 0.459 e 0.650), la discrepanza tra precisione e recall suggerisce che il modello potrebbe essere troppo cauto nel predire queste entità. Per questo motivi, la macro avg ha un F1-score molto basso (0.256), evidenziando che il modello non performa equamente tra tutte le classi, e la weighted avg (0.773) è significativamente più alta, riflettendo la predominanza delle classi più comuni, come "O".

Classe	**Precision**	**Recall**	**F1-score**	**Supporto**
O	0.826	0.973	0.894	3662
B-Organisation	0.668	0.463	0.547	283
I-Organisation	0.680	0.600	0.637	410
B-Temporal	0.875	0.311	0.459	45
I-Temporal	0.929	0.500	0.650	52
B-Nationality	0.000	0.000	0.000	10
B-Location	0.641	0.383	0.480	154
I-Location	0.633	0.326	0.431	190
B-Person	0.854	0.390	0.536	105
I-Person	0.841	0.476	0.608	145
B-DocumentReference	0.000	0.000	0.000	8
I-DocumentReference	0.000	0.000	0.000	20
B-Money	0.000	0.000	0.000	4
I-Money	0.000	0.000	0.000	9
B-Quantity	1.000	0.067	0.125	30
B-MilitaryPlatform	0.000	0.000	0.000	16
I-MilitaryPlatform	0.000	0.000	0.000	26
B-Weapon	0.000	0.000	0.000	17
I-Weapon	0.000	0.000	0.000	14
I-Quantity	0.000	0.000	0.000	32
I-Nationality	0.000	0.000	0.000	2
accuracy			0.805	5234
macro avg	0.378	0.214	0.256	5234
weighted avg	0.772	0.805	0.773	5234

Tabella 4: Classification Report per il modello CRF

La matrice di confusione (Figura 4) fornisce ulteriori dettagli sugli errori di classificazione. Esistono sovrapposizioni tra classi simili. Ad esempio, "B-Organisation" è spesso confusa con "I-Organisation", e lo stesso avviene per altre entità con prefissi "B-" e "I-" (Begin e Inside). Le etichette "B-Location" e "I-Location" mostrano confusione reciproca e anche con classi come "B-Person" o "B-Nationality". Anche le etichette temporali sono confuse, probabilmente a causa della variabilità lessicale o della scarsità di esempi. Gli errori si concentrano principalmente su classi meno rappresentate o ambigue, evidenziando la necessità di aumentare i dati di addestramento per queste entità. La maggior parte delle previsioni corrette sono relative alla classe "O", dato che essa costituisce la maggior parte del dataset. Tuttavia, questo potrebbe mascherare il fatto che il modello fatica con le entità più complesse. La predominanza della classe "O" e la scarsità di dati per altre entità influiscono negativamente sulle performance generali. Il modello non riesce a generalizzare bene per entità con esempi insufficienti.

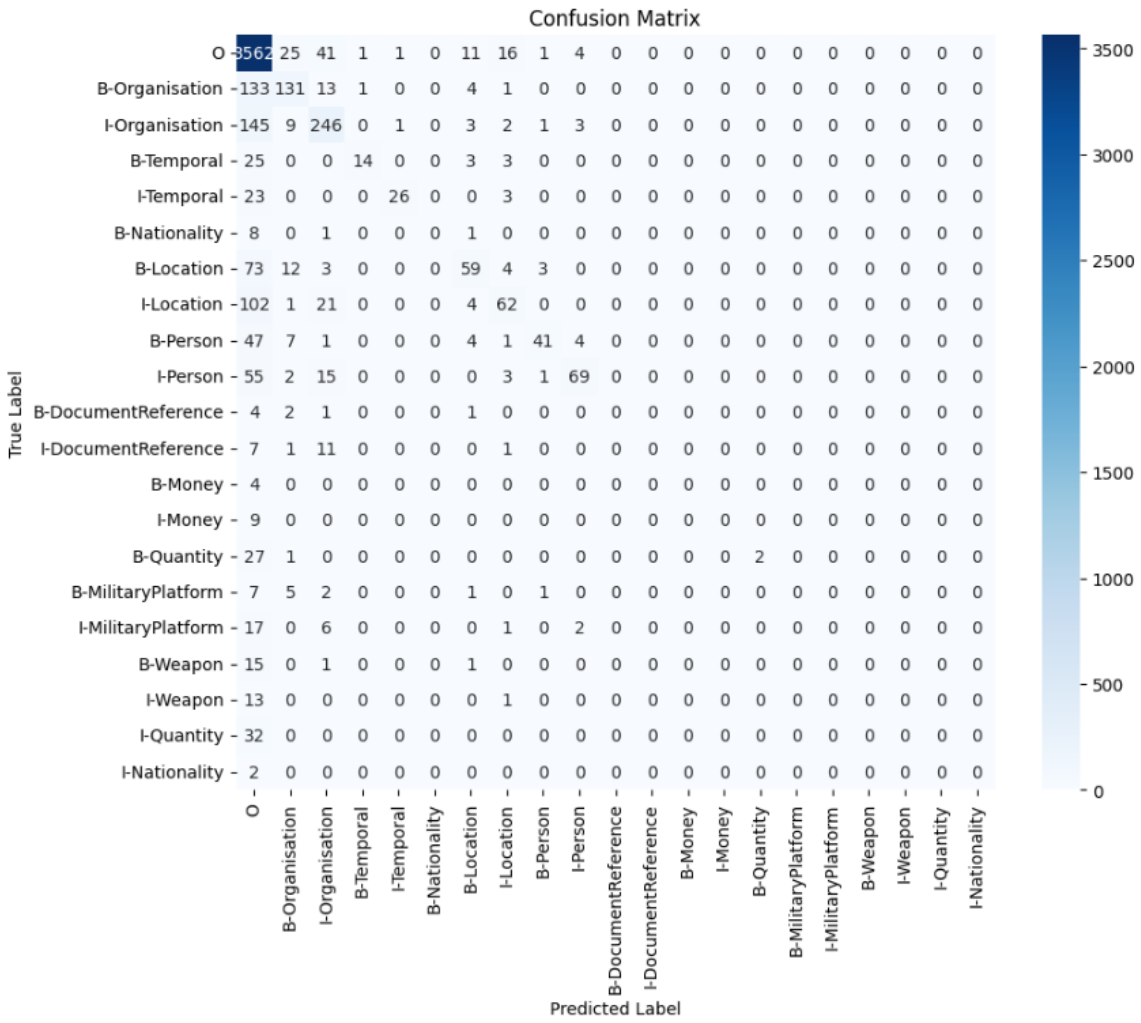


Figura 4: Matrice di confusione per il modello CRF

3.2 BERT

BERT è un modello di linguaggio pre-addestrato, progettato per apprendere rappresentazioni contestuali delle parole attraverso un approccio bidirezionale. Questo lo rende particolarmente adatto a task come il NER, dove il contesto circostante di una parola è cruciale per determinare la sua classe semantica. E' composto da strati di encoder basati sui *Transformer*, e ogni encoder si avvale di meccanismi di attenzione *self-attention* e di *feed-forward* per generare rappresentazioni contestuali. A differenza di modelli come GPT, BERT utilizza il contesto sia a sinistra che a destra di una parola per prevedere parole mascherate (*Masked Language Modeling, MLM*) e per predire la relazione tra coppie di frasi (*Next Sentence Prediction, NSP*). E' inoltre pre-addestrato su grandi corpus di dati generici e successivamente fine-tunato su task specifici. E' infatti disponibile in diverse varianti, tra cui *BERT Base Uncased*, che è una versione con 12 strati, 768 unità nascoste, 12 teste di attenzione e circa 110 milioni di parametri. La denominazione *Uncased* indica che il modello non distingue tra lettere maiuscole e minuscole, un aspetto utile per alcune lingue o task in cui questa differenziazione non è critica. Una specifica applicazione è proprio *BERT for Sequence Classification*, che consente di utilizzare il modello pre-addestrato e adattarlo a compiti specifici come il NER. In questa configurazione, ogni token nella sequenza è classificato in una categoria specifica. Il vantaggio principale di utilizzare BERT per il nostro task è dunque la sua capacità di incorporare informazioni contestuali bidirezionali, contrariamente al precedente approccio (CRF), che necessitava di caratteristiche manuali complesse.

Pipeline

La pipeline da noi progettata per il nostro task si basa proprio su *BERT for Sequence Classification* ed è articolata nelle seguenti fasi principali: **Caricamento e Preprocessing dei Dati**, **Tokenizzazione**, **Preparazione delle Etichette e Creazione delle Maschere di Attenzione**, **Addestramento del Modello**, **Valutazione e Visualizzazione**.
Il caricamento dei dati forniti in formato Conll è analogo al caso precedente. I dati vengono letti e organizzati in una lista di frasi, ciascuna rappresentata da due liste: parole e relative etichette.
Per garantire una dimensione di input uniforme, viene eseguito il padding delle frasi e delle relative etichette, garantendo che tutte le sequenze abbiano la stessa lunghezza, operazione necessaria per l'elaborazione in batch. Il modello richiede inoltre una tokenizzazione specifica per allineare le sequenze di input con il suo vocabolario. La tokenizzazione viene dunque eseguita utilizzando il tokenizer predefinito di BERT, come mostrato nel codice 3.

Listing 3: Tokenizzazione con BertTokenizer

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
#tokenizing the text
tokenized_texts = list(map(lambda x: ['[CLS]'] + tokenizer.tokenize(x) + ['[SEP]'] ,
    ↪ sentences))
```

Ogni frase viene trasformata in una sequenza di token, preceduta dal token speciale `[CLS]` e seguita dal token `[SEP]`.
Le etichette nella terza fase vengono poi convertite in indici numerici per ogni token. Le maschere di attenzione identificano quali token devono essere elaborati. Il codice 4 illustra l'implementazione di queste operazioni.

Listing 4: Preparazione delle etichette e Creazione delle Maschere di attenzione

```
#pre-processing the labels
tags_vals = list(set(label))
tag2idx = {t: i for i, t in enumerate(tags_vals)}

#cutting and padding the tokens and labels to our desired length
input_ids = pad_sequences([tokenizer.convert_tokens_to_ids(txt) for txt in
    ↪ tokenized_texts], maxlen=MAX_LEN, dtype="long", truncating="post", padding="
    ↪ post")
tags = pad_sequences([[tag2idx.get(l, tag2idx['O']) for l in lab] for lab in labels],
    ↪  maxlen=MAX_LEN, value=tag2idx["O"], padding="post", dtype="long", truncating=
    ↪ "post")

#attention masks
attention_masks = [[float(i>0) for i in ii] for ii in input_ids]
```

A questo punto, il modello viene configurato con *BertForTokenClassification* e ottimizzato utilizzando l'algoritmo AdamW (codice 7).

Listing 5: Modeling

```
from transformers import BertForTokenClassification, AdamW

model = BertForTokenClassification.from_pretrained("bert-base-uncased", num_labels=
    ↪ len(tag2idx))

optimizer = AdamW(model.parameters(), lr=3e-5)
```

Il modello viene poi affinato con tecniche di clipping del gradiente e regolarizzazione, e viene addestrato utilizzando i dati di addestramento e valutato sul set di validazione. La funzione di perdita considera la classificazione per ogni token. Infine, la valutazione anche in questo caso viene effettuata calcolando metriche come accuratezza e F1-Score, e attraverso la visualizzazione della matrice di confusione e il classification report.

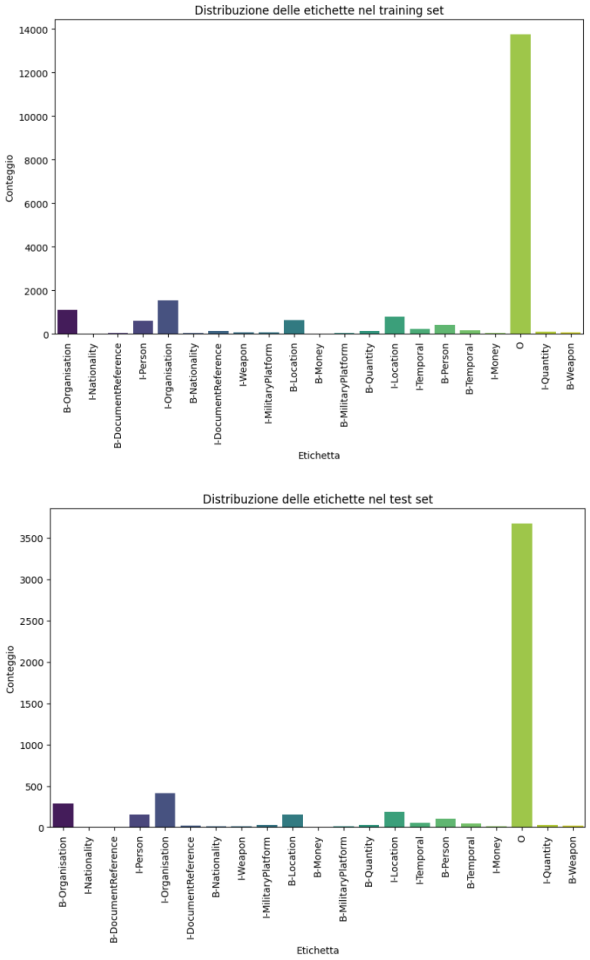
Risultati

Il classification report (Tabella 5), relativo al modello BERT Base Uncased, evidenzia una performance complessiva solida, con un'accuratezza del 91% e una media pesata di precisione, recall e F1-score pari a 0.91. Questo indica che il modello è particolarmente efficace nella gestione delle classi con un supporto maggiore. Tuttavia, come in precedenza, si notano alcune criticità nelle classi con minor supporto o intrinsecamente più difficili, come "B-Money", "B-Nationality" e "B-MilitaryPlatform", dove sia precisione che recall si attestano a 0, riflettendo l'incapacità del modello di identificare correttamente queste entità. Anche per altre classi, come "B-Weapon" e "I-Temporal", le prestazioni sono più limitate, con F1-score inferiori a 0.5, segno che il modello fatica a bilanciare precisione e recall in contesti più complessi o scarsamente rappresentati. Confrontando questi risultati con i risultati ottenuti precedentemente (in Tabella 4), relativi al modello CRF, emergono alcune differenze significative. Sebbene il modello CRF mostri una performance complessivamente inferiore, con un'accuratezza dell'80.5% e una macro media di F1-score pari a 0.256, riesce comunque a ottenere risultati comparabili o superiori in alcune classi specifiche. Ad esempio, la classe "B-Temporal" presenta un F1-score di 0.459 con il CRF, leggermente superiore rispetto al valore di 0.41 ottenuto da BERT Base Uncased. Tuttavia, il modello CRF soffre maggiormente nelle classi meno rappresentate, dove le metriche rimangono a 0. Risulta evidente che il modello BERT Base Uncased beneficia delle sue capacità di generalizzazione, specie per classi con ampio supporto, mentre il CRF risulta meno efficace nella gestione delle classi poco rappresentate e nel bilanciare precisione e recall a livello globale. Dunque, sebbene BERT Base Uncased offre prestazioni superiori e più equilibrate, entrambi i modelli condividono difficoltà comuni con le classi meno rappresentate e risulta ancora chiara la necessità di miglioramento.

Tabella 5: Classification Report per Bert Base Uncased

Label	**Precision**	**Recall**	**F1-Score**	**Support**
B-DocumentReference	0.57	0.36	0.44	11
B-Location	0.52	0.44	0.48	164
B-MilitaryPlatform	0.00	0.00	0.00	24
B-Money	0.00	0.00	0.00	4
B-Nationality	0.00	0.00	0.00	6
B-Organisation	0.42	0.47	0.44	278
B-Person	0.73	0.68	0.70	102
B-Quantity	0.19	0.20	0.19	35
B-Temporal	0.47	0.36	0.41	47
B-Weapon	0.08	0.05	0.06	20
I-DocumentReference	0.79	0.43	0.56	35
I-Location	0.43	0.46	0.45	170
I-MilitaryPlatform	0.24	0.22	0.23	23
I-Money	0.36	0.57	0.44	7
I-Organisation	0.57	0.55	0.56	419
I-Person	0.81	0.76	0.78	162
I-Quantity	0.31	0.45	0.37	20
I-Temporal	0.55	0.35	0.43	65
I-Weapon	0.36	0.40	0.38	10
O	0.95	0.96	0.96	12873
Accuracy			0.91	14475
Macro Avg	0.42	0.39	0.39	14475
Weighted Avg	0.91	0.91	0.91	14475

Il problema principale emerso, già evidenziato nella fase iniziale di Exploratory Data Analysis, è lo sbilanciamento delle entità e, di conseguenza, delle classi presenti nel nostro dataset. Come mostrano i dati riportati in Figura 5, il numero di campioni varia drasticamente tra le diverse classi sia nel Training Set che nel Test Set. Questo squilibrio si traduce in una significativa disparità nella rappresentazione delle classi, con alcune che dominano il dataset (le cosiddette classi maggioritarie) e altre scarsamente rappresentate (le classi minoritarie). In conseguenza a tale sbilanciamento il modello è portato a favorire le classi maggioritarie a discapito di quelle meno rappresentate, e questo comportamento si manifesta con le basse prestazioni osservate fin'ora. Per affrontare questo problema, come verrà presentato in seguito, abbiamo adottato una serie di tecniche di ottimizzazione applicate a una variante più performante del modello BERT. Queste strategie sono state mirate specificamente a migliorare la capacità del modello di gestire lo squilibrio delle classi e garantire una maggiore equità nelle prestazioni, anche per le classi meno rappresentate.



Classe	TrainingS	TestS
B-Organization	1104	284
I-Nationality	2	2
B-DocumentReference	50	8
I-Person	605	150
I-Organization	1539	411
B-Nationality	34	20
I-DocumentReference	146	10
I-Weapon	70	14
I-MilitaryPlatform	64	26
B-Location	644	155
B-Money	20	4
B-MilitaryPlatform	47	16
B-Quantity	150	30
I-Location	784	191
I-Temporal	225	55
B-Person	411	106
B-Temporal	177	46
I-Money	36	9
O	13747	3672
I-Quantity	94	32
B-Weapon	81	17

Figura 5: Distribuzione delle etichette nei dataset di Training e di Test

3.3 RoBERTa

Per poter migliorare le prestazioni del nostro modello, abbiamo scelto innanzitutto di utilizzare *RoBERTa*, acronimo di *Robustly Optimized BERT Approach*. RoBERTa rappresenta un'evoluzione del modello *BERT (Bidirectional Encoder Representations from Transformers)* che si concentra sull'ottimizzazione delle tecniche di pre-addestramento per migliorare le prestazioni sui compiti di *NLP (Natural Language Processing)*. Introdotto da *Facebook AI* nel 2019, RoBERTa mantiene l'architettura Transformer bidirezionale di BERT, ma apporta modifiche significative al processo di addestramento. In particolare, rimuove il compito di previsione della prossima frase (*Next Sentence Prediction - NSP*), che si era dimostrato poco influente nelle prestazioni di BERT, e si concentra esclusivamente sul compito di *Masked Language Modeling (MLM)*, in cui alcune parole nell'input vengono mascherate e il modello deve prevederle. Questa scelta consente di sfruttare pienamente il suo potenziale computazionale per imparare rappresentazioni linguistiche più robuste. Un altro miglioramento chiave è l'utilizzo di dataset significativamente più grandi e diversificati, inclusi *CommonCrawl News*, *OpenWebText*, *BooksCorpus* e *Wikipedia*. Utilizza anche sequenze di input più lunghe e batch di dimensioni maggiori, il che migliora l'efficienza dell'addestramento e consente al modello di apprendere modelli contestuali più complessi. Inoltre, implementa una regolazione dinamica della probabilità di mascheramento durante l'addestramento, rendendo il processo più stabile e accurato.
I vantaggi di RoBERTa rispetto a BERT sono evidenti in termini di prestazioni. Grazie all'ottimizzazione delle tecniche di pre-addestramento, RoBERTa supera BERT in quasi tutti i benchmark standard di NLP, come *GLUE*, *SQuAD* e *RACE*. La rimozione del compito NSP elimina un potenziale collo di bottiglia computazionale, mentre l'uso di dataset più ampi e il training più lungo permettono a RoBERTa di generalizzare meglio rispetto a BERT, soprattutto su compiti con dati complessi o di

grandi dimensioni. Tuttavia, queste migliorie hanno un costo: RoBERTa richiede infatti risorse computazionali significativamente maggiori per l'addestramento, rendendolo più impegnativo dal punto di vista infrastrutturale rispetto a BERT. Nonostante ciò, rappresenta un passo avanti cruciale nel campo del pre-addestramento per modelli di linguaggio, consolidandosi come uno degli approcci più robusti e performanti nella moderna NLP.

Pipeline

Descriviamo ora la pipeline implementata per il nostro task utilizzando il modello pre-addestrato RoBERTa. Come in precedenza, la pipeline si compone delle usuali fasi principali: **preparazione dei dati**, **Tokenizzazione e Allineamento delle Etichette**, **Definizione del Modello**. **Addestramento del modello**, **Valutazione**. Il framework `Hugging Face Transformers` è stato utilizzato per sfruttare le capacità di RoBERTa per il task di token classification.
Anche qui, dopo il caricamento dei dati in formato CoNLL, le etichette testuali vengono mappate su indici numerici per adattarsi all'input richiesto dal modello. Come per BERT, anche per utilizzare RoBERTa è necessario tokenizzare i dati. Tuttavia, poiché RoBERTa utilizza la *tokenizzazione subword*, è importante allineare le etichette originali dei token ai token generati dal tokenizzatore. Questo viene fatto introducendo etichette `-100` per ignorare i token subword aggiuntivi attraverso la funzione mostrata nel codice 6, che assicura che il modello apprenda solo dai token principali di ogni parola, ignorando i token subword generati automaticamente.

Listing 6: Tokenizzazione RoBERTa

```
# Initialize the tokenizer with add_prefix_space=True
tokenizer = AutoTokenizer.from_pretrained(model_name, add_prefix_space=True)

def tokenize_and_align_labels(examples):
    tokenized_inputs = tokenizer(
        examples["tokens"], truncation=True, is_split_into_words=True, padding="
    ↪ max_length", max_length=128
    )
    labels = []
    for i, label in enumerate(examples["tags"]):
        word_ids = tokenized_inputs.word_ids(batch_index=i)
        label_ids = []
        previous_word_id = None
        for word_id in word_ids:
            if word_id is None:
                label_ids.append(-100)  # Ignora padding
            elif word_id != previous_word_id:  # Primo token della parola
                label_ids.append(label[word_id])
            else:  # Token aggiuntivi
                label_ids.append(-100)
            previous_word_id = word_id
        labels.append(label_ids)
    tokenized_inputs["labels"] = labels
    return tokenized_inputs

tokenized_datasets = dataset.map(tokenize_and_align_labels, batched=True)
```

Il modello RoBERTa (`roberta-base`) viene caricato e adattato al task di NER definendo il numero di etichette richiesto dal dataset. La configurazione del modello mappa inoltre le etichette numeriche ai nomi delle classi per facilitare l'interpretazione dei risultati (codice 7):

Listing 7: Definizione del modello

```
from transformers import AutoModelForTokenClassification

model_name = "roberta-base"
model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(
    ↪ unique_tags))
model.config.id2label = id2tag
```

```
model.config.label2id = tag2id
```

L'addestramento è gestito utilizzando la classe `Trainer` del framework `Hugging Face Transformers`, che semplifica il processo. Gli argomenti di configurazione specificano ad esempio il numero di epoche, il batch size, la strategia di salvataggio e il monitoraggio della metrica `F1`, e sono mostrati nel seguente codice 8:

Listing 8: Configurazione del Trainer

```
from transformers import TrainingArguments, Trainer

training_args = TrainingArguments(
output_dir="./results",
evaluation_strategy="epoch",
learning_rate=2e-5,
per_device_train_batch_size=16,
per_device_eval_batch_size=16,
num_train_epochs=5,
weight_decay=0.01,
logging_dir="./logs",
logging_steps=10,
save_total_limit=2,
load_best_model_at_end=True,
metric_for_best_model="f1",
greater_is_better=True,
save_strategy="epoch",
)

trainer = Trainer(
model=model,
args=training_args,
train_dataset=tokenized_datasets["train"],
eval_dataset=tokenized_datasets["test"],
tokenizer=tokenizer,
compute_metrics=compute_metrics,
)
```

Dopo aver configurato il Trainer il modello viene addestrato, e al termine dell'addestramento, il modello viene valutato sul set di test. Le principali metriche calcolate, anche in questo caso, includono Precision, Recall, F1-score e Accuracy complessiva.
Grazie alle precedenti tecniche utilzzate nella pipeline, è stato possibile ottenere risultati accurati e sfruttare appieno le capacità di questo modello pre-addestrato di ultima generazione.

Risultati

Le prestazioni del modello RoBERTa, valutate sulla base dei risultati forniti in Tabella 6 e in Figura 6, mostrano un comportamento complessivamente positivo, con un'accuratezza generale pari all'89% e un F1-score medio ponderato di 0.88. Questo indica che il modello è in grado di effettuare previsioni affidabili, soprattutto per le etichette più frequenti e ben rappresentate nel dataset. Tuttavia, un'analisi più dettagliata mette in luce alcune aree di forza, ma anche diverse debolezze. Il modello eccelle nel riconoscere entità comuni come le persone ("B-Person" e "I-Pers on") e le organizzazioni ("I-Organisation"), ottenendo rispettivamente F1-score di 0.88 e 0.79. Queste categorie hanno un numero significativo di esempi nel dataset, permettendo al modello di apprendere le loro caratteristiche con maggior precisione. Anche le entità temporali, come "B-Temporal" e "I-Temporal", sono gestite con buone performance, a dimostrazione della capacità del modello di distinguere informazioni relative al tempo. Nonostante queste ottime prestazioni, anche RoBERTa incontra difficoltà con le etichette meno rappresentate, come "B-MilitaryPlatform" o "B-Nationality", per le quali l'F1-score è pari a zero. Questo comportamento è attribuibile alla scarsità di esempi disponibili per queste categorie durante l'addestramento, che impedisce al modello di apprendere pattern significativi. La matrice di confusione conferma inoltre la presenza di ambiguità tra etichette simili, come "B-Location" e "I-Location" o tra "B-Organisation" e "I-Organisation", suggerendo che il modello fatichi a distinguere correttamente tra token appartenenti a

diverse posizioni grammaticali della stessa entità. Nel complesso, anche le metriche macro medie, come la precisione e il richiamo (pari a circa 0.50), riflettono un comportamento non uniforme tra le diverse categorie: il modello tende a favorire le etichette più frequenti, penalizzando quelle con minor supporto.

Tabella 6: RoBERTa test 1: Classification Report

Label	Precision	Recall	F1-Score	Support
B-DocumentReference	0.00	0.00	0.00	8
B-Location	0.79	0.79	0.79	155
B-MilitaryPlatform	0.00	0.00	0.00	16
B-Money	0.00	0.00	0.00	4
B-Nationality	0.00	0.00	0.00	10
B-Organisation	0.74	0.80	0.77	284
B-Person	0.82	0.96	0.88	106
B-Quantity	0.68	0.83	0.75	30
B-Temporal	0.69	0.76	0.72	46
B-Weapon	0.33	0.06	0.10	17
I-DocumentReference	0.60	0.75	0.67	20
I-Location	0.71	0.89	0.79	191
I-MilitaryPlatform	0.00	0.00	0.00	26
I-Money	1.00	0.11	0.20	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.72	0.88	0.79	411
I-Person	0.85	0.82	0.83	150
I-Quantity	0.84	0.84	0.84	32
I-Temporal	0.69	0.89	0.78	55
I-Weapon	0.75	0.43	0.55	14
O	0.96	0.93	0.94	3672
accuracy			0.89	5258
macro avg	0.53	0.51	0.50	5258
weighted avg	0.88	0.89	0.88	5258

Confrontando i risultati di RoBERTa con quelli di BERT Base Uncased (Tabella 5), si osserva come quest'ultimo ottenga un'accuratezza leggermente superiore (91%) e un F1-score ponderato di 0.91, ma presenti prestazioni complessivamente inferiori su alcune categorie. BERT, infatti, si comporta peggio per etichette come "B-Temporal", "B-Organisation" e "B-Person", in cui RoBERTa riesce a distinguersi per precisione e richiamo. Tuttavia, BERT mostra una distribuzione delle performance più uniforme, con meno differenze tra etichette comuni e meno rappresentate. Questo potrebbe derivare da una maggiore tendenza del modello a trattare le classi con equilibrio, pur sacrificando in parte le prestazioni sulle etichette dominanti. Mentre BERT presenta un alto livello di confusione su etichette rare, RoBERTa dimostra una maggiore capacità di distinguere alcune entità meno frequenti, come le informazioni temporali o i riferimenti a documenti. Dunque, RoBERTa emerge come il modello più robusto, soprattutto per le entità più comuni e ben rappresentate, e mostra un vantaggio nei casi in cui è richiesto un riconoscimento più fine. Infatti, sebbene BERT mostri prestazioni più uniformi, RoBERTa risulta superiore per accuratezza e capacià di generalizare su entità frequenti e temporalmente rilevanti. Tuttavia, entrambi i modelli soffrono in presenza di etichette scarsamente supportate, evidenziando la necessità di strategie migliorative come il riequilibrio delle classi, l'aumento dei dati di addestramento o l'impiego di modelli ensemble.

3.4 Ottimizzazione

Per superare il problema dello sbilanciamento delle classi, sul modello RoBERTa abbiamo adottato in maniera incrementale una serie di strategie di ottimizzazione. Queste hanno riguardato aspetti chiave come l'utilizzo di pesi per le classi meno frequenti, l'applicazione di varie tecniche di data augmentation e di augmentations ricorsive, e lo studio di quali classi specifihe sottoporre maggiormente alla data augmentation. Abbiamo inoltre aumentato il numero di epoche per garantire i migliori risultati senza cadere nell'errore di *overfitting*, e testato una loss function personalizzata. I test di ottimizzazione progressivi che abbiamo effettuato a partire dall'addestramento precedente sul modello RoBERTa (test 1) sono esposti nella Tabella 7.

Una tecnica utilizzata nel test 3 per far venire meno il problema dello sbilanciamento del dataset, è stata il calcolo di un peso per ciascuna classe (*Class Weights*) basandosi sulla frequenza delle etichette nel set di addestramento. Assegnando infatti nella funzione di perdita un peso maggiore alle classi meno rappresentate, si aiuta il modello a migliorare la sensibilità verso queste entità. I pesi vengono calcolati utilizzando la funzione `compute_class_weight` del modulo `sklearn` (codice 9). Tali pesi vengono applicati andando a definire una (`Focal Loss`) (codice 10), progettata per gestire lo squilibrio tra classi concentrando l'attenzione sugli errori più difficili da correggere, ottenendo però risultati più

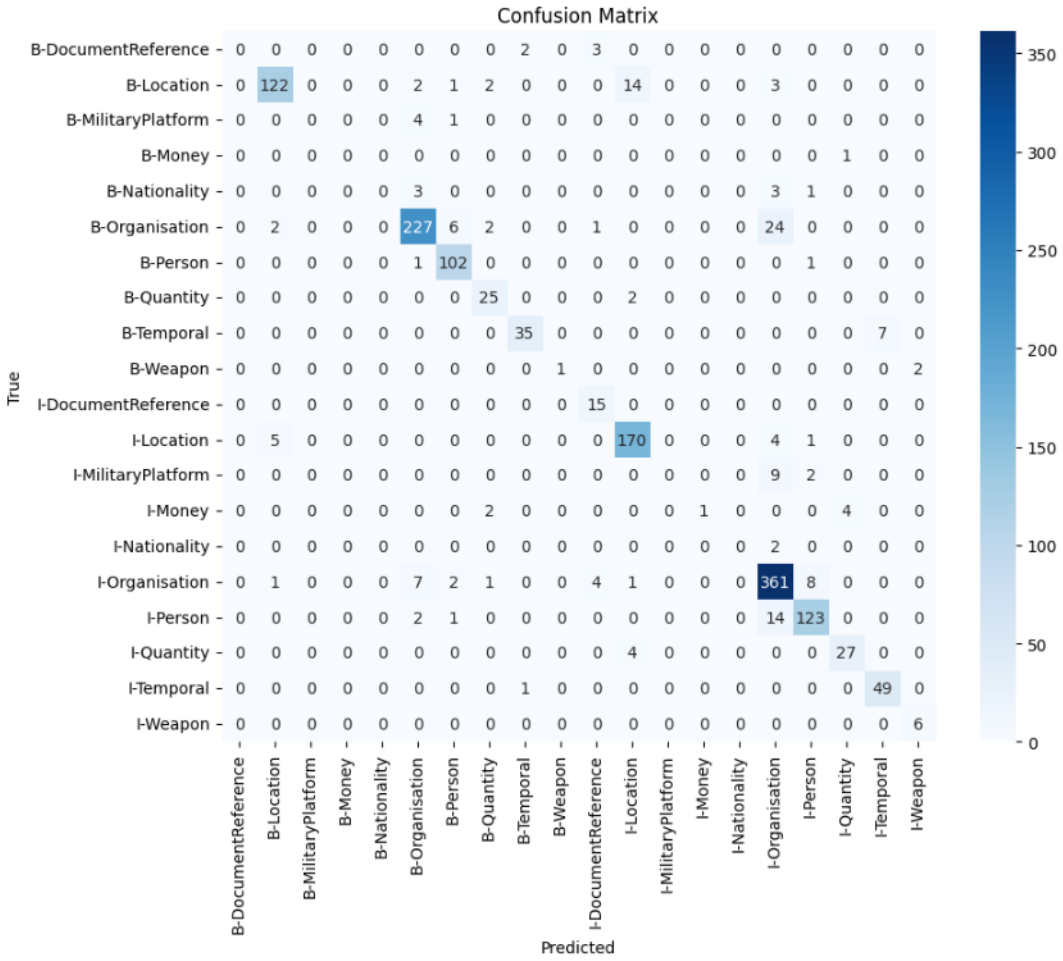


Figura 6: RoBERTa test 1: Matrice di Confusione

Tabella 7: Ottimizzazioni testate sul modello RoBERTa

N.	**Model**	**C.Weights**	**Data Augmentation**	**Loss**	**Epochs**
1	RoBERTa base	No	no	CrossEntropy	5
2	RoBERTa base	No	**S (3 classes, 2 aug)**	CrossEntropy	5
3	RoBERTa base	**yes**	S (3 classes, 2 aug)	**Focal**	**10**
4	RoBERTa base	No	S (5 classes, **5 aug**)	CrossEntropy	**11**
5	RoBERTa base	No	**S,CE,RC** (5 classes, 2 aug)	CrossEntropy	11
6	RoBERTa base	No	S, CE, RC (5 classes, **4 aug**)	Cross Entropy	**9**
7	**RoBERTa large**	No	S, CE, RC (5 classes, 2 aug)	Cross Entropy	11

Legenda: S = Synonym Aug, CE = Contextual Embedding Aug, RC = Random Character Aug

scarsi rispetto alla classica Cross Entropy Loss.
Per arricchire il dataset e migliorare la capacità del modello di generalizzare su dati non visti, abbiamo poi applicato diverse tecniche di data augmentation e abbiamo regolato il numero di augmentations ricorsive da applicare alle classi scelte. Le tipologie di augmentations applicate sono le seguenti:

- **Synonym Augmentation (S)**: Consiste nella sostituzione di parole con loro sinonimi. È stata applicata a un numero crescente di classi problematiche sulla base dei risultati ottenuti, partendo da 3 nei primi tentativi, fino a 5 negli ultimi, nei quali è stata combinata anche alle seguenti tecniche.
- **Contextual Embedding Augmentation (CE)**: Seleziona sinonimi o sostituzioni contestual-

mente appropriati utilizzando embedding contestuali, preservando il significato delle frasi.

- **Random Character Augmentation (RC)**: Introduce errori casuali a livello di caratteri, come inserimenti, cancellazioni o sostituzioni. È utile per simulare errori nei dati reali.

La funzione `augment_data` nel codice 11 mostra come abbiamo applicato in congiunto i tre tipi di data augmentation principalmente alle classi più problematiche del nostro dataset di training.

Listing 9: Class Weights

```
labels = [tag for tags in train_tag_ids for tag in tags]
class_weights = compute_class_weight(class_weight="balanced", classes=np.unique(
    ↪ labels), y=labels)
class_weights = torch.tensor(class_weights, dtype=torch.float)
```

Listing 10: Focal Loss

```
class FocalLoss(nn.Module):
    def __init__(self, alpha=None, gamma=2, ignore_index=-100):
        super(FocalLoss, self).__init__()
        self.alpha = alpha
        self.gamma = gamma
        self.ignore_index = ignore_index

    def forward(self, logits, labels):
        ce_loss = nn.CrossEntropyLoss(weight=self.alpha, ignore_index=self.
    ↪ ignore_index)(logits.view(-1, logits.size(-1)), labels.view(-1))
        pt = torch.exp(-ce_loss)
        focal_loss = (1 - pt) ** self.gamma * ce_loss
        return focal_loss.mean()
```

Listing 11: Data Augmentation

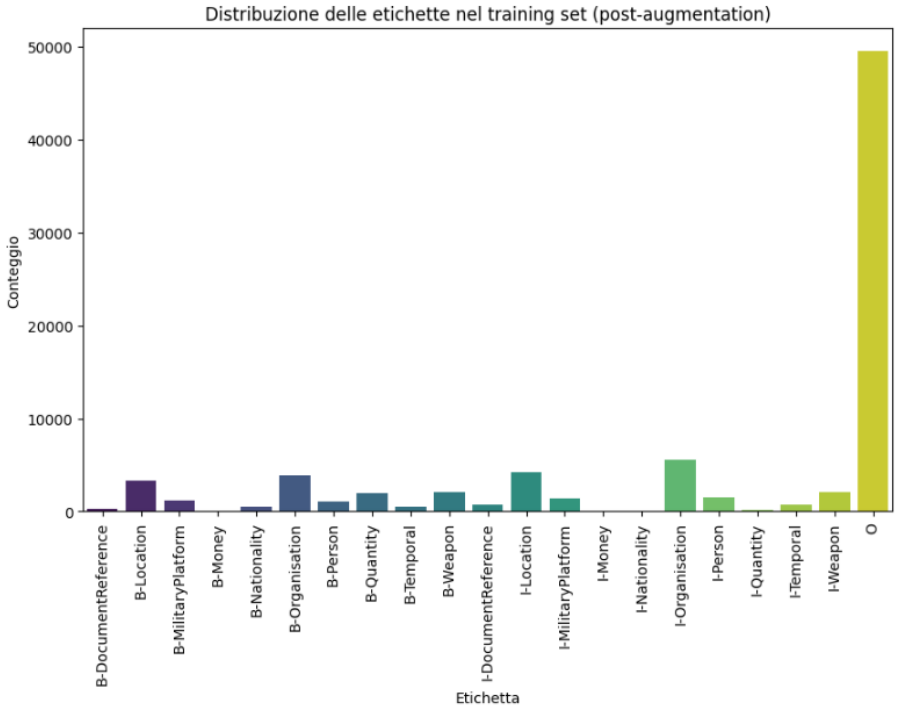
```
def augment_data(dataset, target_label, n_augment=5):
    aug_synonyms = naw.SynonymAug(aug_src='wordnet', aug_p=0.3)
    aug_contextual = naw.ContextualWordEmbsAug(model_path='bert-base-uncased', aug_p
    ↪ =0.2)
    aug_scramble = nac.RandomCharAug(action='swap', aug_char_min=2, aug_word_p=0.2)
    augmented_data = []
    for example in dataset:
        if target_label in example["tags"]:
            for _ in range(n_augment):
                # Applica tecniche di augmentation
                augmented_text_1 = aug_synonyms.augment(example["tokens"])
                augmented_text_2 = aug_contextual.augment(example["tokens"])
                augmented_text_3 = aug_scramble.augment(example["tokens"])

                augmented_data.extend([
                    {"tokens": augmented_text_1, "tags": example["tags"]},
                    {"tokens": augmented_text_2, "tags": example["tags"]},
                    {"tokens": augmented_text_3, "tags": example["tags"]},
                ])
    return augmented_data

low_freq_labels = ["B-MilitaryPlatform", "B-Nationality", "B-Weapon", "I-Nationality"
    ↪ , "I-Weapon"]
augmented_train = []
for label in low_freq_labels:
    augmented_train += augment_data(train_data, tag2id[label], n_augment=2)

train_data += augmented_train
dataset["train"] = Dataset.from_list(train_data)
```

La nuova distribuzione delle classi nel dataset di training in seguito all'applicazione della data augmentation è illustrata in Figura 7.



Classe	TSet
B-DocumentRef	266
B-Loc	3284
B-MilitaryPlat	1199
B-Money	56
B-National	526
B-Org	3888
B-Person	1035
B-Quant	2010
B-Temp	549
B-Weapon	2085
I-DocumentRef	674
I-Location	4204
I-MilitaryPlatf	1420
I-Money	60
I-National	50
I-Org	5619
I-Pers	1493
I-Quant	214
I-Temp	681
I-Weapon	2098
O	49483

Figura 7: Distribuzione delle etichette nei dataset di Training in seguito alla data augmentation

Partendo dai primi tentativi nei quali il modello è stato addestrato con 5 epoche, monitorando i valori della loss di training in relazione alla loss di validation, abbiamo aumentato il numero fino a trovare il giusto tradeoff che consentisse al modello di apprendere in modo più approfondito i pattern presenti nei dati, senza però arrivare a commettere un errore di overfitting e compromettere le capacità di generalizzazione del modello. Il tradeoff è stato individuato in un numero di epoche compreso tra 9 e 11, in corrispondenza del quale la loss di validazione cominciava a mostrare i primi cenni di aumento. Nel test 7 inoltre, abbiamo testato la variante large del modello RoBERTa (`roberta-large`), non ottenendo però particolari miglioramenti rispetto ai risultati ottenuti con la variante di base (`roberta-base`). I risultati ottenuti nei test effettuati sono riportati in Tabella 8. Il miglior modello è quello corrispondente al test 5, raggiungendo i migliori valori complessivi nelle metriche di performance, e dimostrando una notevole capacità di generalizzazione e una solida gestione delle classi meno rappresentate.
Nel test in questione è stata utilizzata la versione di base di RoBERTa, e sono state applicate ricorsivamente per 2 volte tutte le tecniche di data augmentation (Synonym, Contextual Embedding, Random Character), addestrando il modello per 11 epoche e con la classica Cross Entropy Loss.
Le ottimizzazioni implementate, applicate in maniera incrementale, hanno dimostrato dunque l'importanza di combinare tecniche di data augmentation e una configurazione ottimale di epoche e augmentations ricorsive. Il processo progressivo ci ha permesso di valutare l'impatto di ogni intervento e di individuare la configurazione che garantisse prestazioni superiori nel task di NER. Analizziamo ora in dettaglio i risultati definitivi ottenuti dalla migliore implementazione, corrispondente al test 5.

Risultati

L'andamento del training mostrato nella tabella in Figura 8 evidenzia un processo di apprendimento progressivo e ben strutturato, con una chiara riduzione delle perdite (loss) e un miglioramento costante delle metriche di valutazione principali. Si osserva infatti una significativa riduzione della Training Loss

Tabella 8: Risultati dei test di ottimizzazione del modello RoBERTa

N.	evLoss	evPr	evRec	evF1	evAcc	mavgPr	mavgRec	mavgF1	wavgPr	wavgRec	wavgF1
1	0.382	0.620	0.722	0.667	0.887	0.53	0.51	0.50	0.88	0.89	0.88
2	0.344	0.622	0.768	0.687	0.981	0.61	0.60	0.58	0.90	0.89	0.89
3	**0.157**	0.398	0.714	0.511	0.791	0.46	**0.83**	0.56	0.88	0.79	0.81
4	0.41	0.679	0.786	0.729	0.900	0.72	0.74	**0.73**	0.90	**0.90**	**0.90**
5	0.380	0.680	**0.803**	0.736	0.900	0.71	0.75	0.72	**0.91**	**0.90**	**0.90**
6	0.380	0.680	0.790	0.731	**0.902**	**0.73**	0.73	0.72	**0.91**	**0.90**	**0.90**
7	0.457	**0.688**	0.772	0.728	0.896	0.72	0.71	0.71	0.90	**0.90**	**0.90**

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	1.020500	0.518538	0.502494	0.596154	0.545332	0.845378
2	0.753800	0.369828	0.590024	0.717456	0.647530	0.882655
3	0.655500	0.334286	0.621053	0.785503	0.693664	0.892925
4	0.494300	0.329515	0.650544	0.795858	0.715902	0.899772
5	0.473600	0.317883	0.655087	0.781065	0.712551	0.901293
6	0.457700	0.322804	0.692206	0.775148	0.731333	0.905287
7	0.381200	0.339975	0.672131	0.788462	0.725664	0.901483
8	0.364300	0.358607	0.674157	0.798817	0.731212	0.900152
9	0.326200	0.379783	0.679599	0.803254	0.736271	0.899582
10	0.224100	0.381638	0.666667	0.792899	0.724324	0.901103
11	0.316200	0.382137	0.668766	0.785503	0.722449	0.901674

Figura 8: RoBERTa test 5: andamento del training

e della Validation Loss nelle prime epoche, a indicare che il modello sta apprendendo le caratteristiche del dataset, migliorando sia nelle prestazioni su dati visti durante l'addestramento che nella generalizzazione sui dati di validazione. Nelle ultime epoche, il calo della loss diventa più lento e raggiunge una sorta di plateau, segnalando che il modello si sta avvicinando a un punto di convergenza. La Precision mostra un costante incremento iniziale, migliorando da 0.50 nella prima epoca a 0.67 nell'undicesima. Questo implica che il modello riduce progressivamente i falsi positivi. La Recall cresce rapidamente nelle prime epoche e si stabilizza successivamente intorno a 0.78-0.80, suggerendo che il modello identifica efficacemente la maggior parte dei campioni positivi. L'F1-Score, che combina Precision e Recall, segue un andamento simile, passando da 0.54 a un valore massimo di 0.73 nella sesta epoca, per poi stabilizzarsi poco sotto. Questo andamento conferma che il modello migliora il bilanciamento tra falsi positivi e falsi negativi. Anche l'Accura-

Figura 9: RoBERTa test 5: monitoraggio delle metriche con wandb

cy mostra un incremento rapido e consistente inizialmente, per poi raggiungere un valore di circa 0.90 già alla sesta epoca, con ulteriori piccole oscillazioni nelle successive. Dunque, il training è

stato molto efficace, con una convergenza stabile e metriche di valutazione che dimostrano un buon equilibrio tra accuratezza e capacità di generalizzazione. Per monitorare con precisione l'andamento delle metriche e le variazioni dei valori nel corso degli epoche, abbiamo utilizzato *Weights & Biases (wandb)*, una piattaforma di monitoraggio e gestione degli esperimenti che consente di tracciare automaticamente metriche, parametri di configurazione e grafici relativi ai modelli di machine learning durante il processo di training e validazione. Wandb ci ha permesso di visualizzare in tempo reale le curve di andamento delle metriche principali, e confrontare facilmente i vari esperimenti effettuati, migliorando significativamente il processo di analisi. In Figura 9 sono illustrati alcuni dei grafici delle evaluation metrics prodotti da wandb, relativi al nostro miglor modello. In basso, nelle Figure 10 e 11, troviamo rispettivamente l'andamento dell'F1-score nel corso delle epoche del test 5, e un confronto tra quest'ultimo e gli andamenti degli F1-scores degli altri test effettuati. I risultati rappresentati nel classification report in Tabella 9 presentano un ottimo quadro generale, con performance solide nelle classi principali e significativi miglioramenti rispetto a situazioni di totale mancanza di riconoscimento per alcune categorie. Si osserva una buona capacità del modello di distinguere le entità più frequenti, come "B-Location", "B-Organisation" e "B-Person", con F1-score che raggiungono o superano 0.80, dimostrando una combinazione bilanciata di precision e recall. Questo evidenzia che il modello riesce a identificare correttamente la maggior parte delle occorrenze e a limitare gli errori di classificazione. Anche le classi con supporto moderato, come B-Temporal e B-Quantity, mostrano risultati promettenti, con F1-score rispettivamente di 0.82 e 0.80, segno che il modello è efficace anche in contesti più variabili. L'accuratezza generale aumenta al 90%, che indica che il modello ha una solida capacità di classificazione a livello complessivo. Permangono comunque alcune difficoltà per le categorie con basso supporto, come "B-Money", "B-MilitaryPlatform" e "I-Nationality", dove gli F1-score rimangono bassi, ma confrontando questi risultati con quelli precedenti (Tabella 6), i

Tabella 9: RoBERTa test 5: Classification Report

Label	Precision	Recall	F1-Score	Support
B-DocumentReference	0.86	0.75	0.80	8
B-Location	0.75	0.86	0.80	155
B-MilitaryPlatform	0.43	0.56	0.49	16
B-Money	0.67	0.50	0.57	4
B-Nationality	0.83	0.50	0.62	10
B-Organisation	0.76	0.80	0.78	284
B-Person	0.88	0.94	0.91	106
B-Quantity	0.70	0.93	0.80	30
B-Temporal	0.74	0.91	0.82	46
B-Weapon	0.48	0.59	0.53	17
I-DocumentReference	0.64	0.80	0.71	20
I-Location	0.75	0.92	0.82	191
I-MilitaryPlatform	0.78	0.69	0.73	26
I-Money	0.88	0.78	0.82	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.78	0.87	0.82	411
I-Person	0.89	0.89	0.89	150
I-Quantity	0.97	0.94	0.95	32
I-Temporal	0.78	0.93	0.85	55
I-Weapon	0.44	0.57	0.50	14
O	0.96	0.92	0.94	3672
accuracy			0.90	5258
macro avg	0.71	0.75	0.72	5258
weighted avg	0.91	0.90	0.90	5258

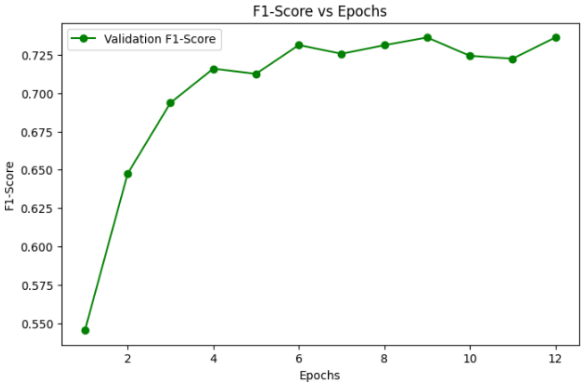


Figura 10: Descrizione immagine 1

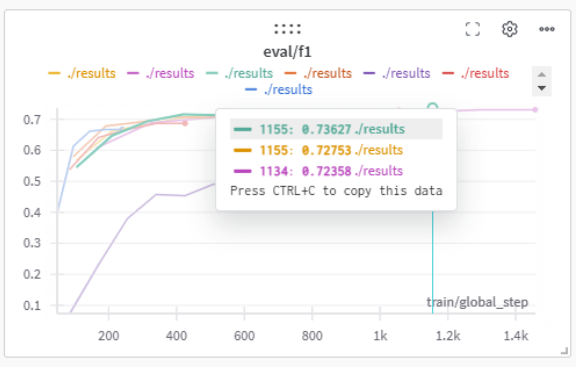


Figura 11: Descrizione immagine 2

miglioramenti sono evidenti e sostanziali. Il macro F1-score passa da 0.50 a 0.72, evidenziando una maggiore capacità del modello di trattare anche le classi meno comuni. Ad esempio, per la classe "B-DocumentReference", si registra un passaggio da un F1-score di 0.00 a 0.80, dimostrando che il modello ora riesce a identificare efficacemente questa entità. Analogamente, "B-MilitaryPlatform", che in precedenza non veniva riconosciuta, ora presenta un F1-score di 0.49. Anche "B-Nationality" mostra un miglioramento significativo, passando da uno score nullo a 0.62. Questi dati confermano che il modello non solo è migliorato nel riconoscimento delle classi più frequenti, ma ha anche acquisito capacità nelle categorie più difficili. Alcune debolezze, tuttavia, persistono, come nel caso di "I-Nationality", che continua a non essere riconosciuta, e di "I-Weapon", che pur mostrando qualche progresso rimane problematica con un F1-score di 0.50. Rispetto ai risultati iniziali, questi aspetti rappresentano ancora un limite, ma il miglioramento complessivo suggerisce che il modello sta evolvendo nella giusta direzione. La capacità di mantenere elevate le performance per le classi frequenti, come B-Location e B-Organisation, mentre si potenziano le prestazioni nelle classi meno comuni, dimostra che il modello è diventato più bilanciato e affidabile, come è possibile osservare anche nella matrice di confusione in Figura 12.

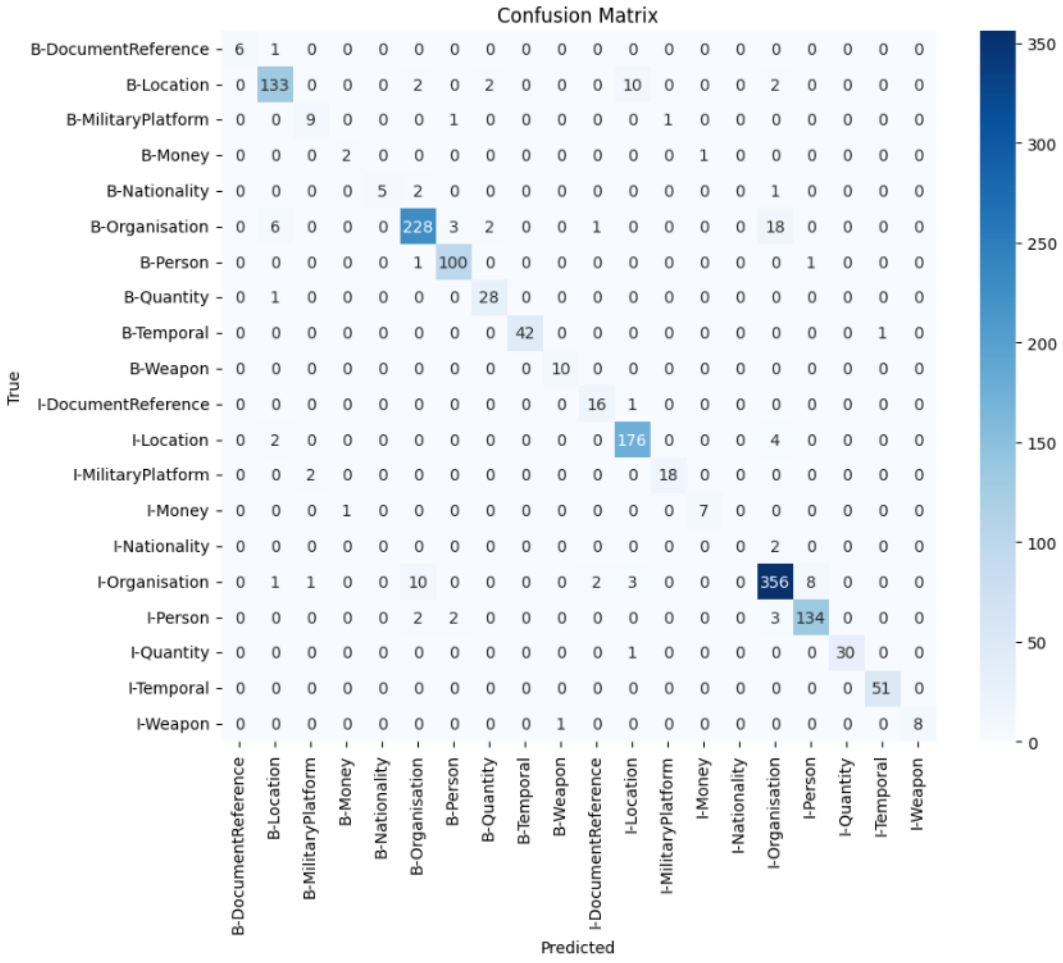


Figura 12: RoBERTa test 5: Matrice di Confusione

4 Conclusioni

In questo lavoro abbiamo dunque esplorato l'utilizzo di varie tecniche per il task di Named Entity Recognition (NER) e in particolare l'ottimizzazione del modello RoBERTa, concentrandoci sull'applicazione di tecniche di data augmentation e sul tuning dei parametri di addestramento. Dopo aver condotto una serie di esperimenti, il miglior modello è stato individuato nel test 5, che ha raggiunto significativi

miglioramenti nelle metriche di performance rispetto ai test precedenti e successivi. In particolare, l'utilizzo della versione di base di RoBERTa, combinato con due cicli di data augmentationn di tre tipi diversi, ha permesso di ottenere un bilanciamento tra le performance su dati di addestramento e la capacità di generalizzazione sui dati di validazione, riducendo al minimo il rischio di overfitting. I risultati ottenuti, con un F1-score che supera 0.80 per molte delle classi principali, dimostrano l'efficacia dell'approccio utilizzato, che ha migliorato significativamente anche il riconoscimento delle entità meno rappresentate nel dataset, centrando dunque il nostro obbiettivo.
In prospettiva, futuri lavori potrebbero concentrarsi sull'esplorazione di altre tecniche di data augmentation, come la *back-translation*, un *Hyperparameter tuning* completo, o l'utilizzo di modelli preaddestrati su domini specifici. Inoltre, sarebbe interessante sperimentare con modelli ancora più avanzati o esplorare diverse architetture di rete per migliorare ulteriormente le prestazioni. Nonostante alcune aree di miglioramento, i risultati complessivi suggeriscono che il modello sviluppato ha un ottimo potenziale per il riconoscimento delle entità in contesti complessi, confermando l'efficacia delle ottimizzazioni adottate.

5 Note finali

Per lo sviluppo del nostro progetto, abbiamo eseguito i nostri esperimenti su *Google Colab*, sfruttando le GPU disponibili sulla piattaforma. Questa scelta ci ha permesso di condurre test e allenamenti in modo efficiente, mantenendo al contempo una compatibilità universale del progetto. In dettaglio, è possibile trovare tali scripts nella repository di progetto [ZM24]. Al suo interno sono disponibili i notebooks corrispondenti alla fase di EDA e allo sviluppo del task utilizzando le tre metodologie: CRF, BERT, RoBERTa. I notebooks riguardanti RoBERTa in particolare sono numerati coerentemente ai dati della Tabella 7, e al loro interno è possibile trovare i risultati specifici di ogni test effettuato, a partire dall'andamento del training fino alle matrici di confusione e altri grafici di supporto. In questo modo rendiamo possibile il confronto diretto tra le varie tecniche adottate, per poterle eventualmente recuperare in seguito e sperimentare con ulteriori test.

Appendice A

Dati aggiuntivi

In questa sezione sono stati inclusi ulteriori dati a supporto della comprensione e della valutazione dei risultati ottenuti nel lavoro. Tra questi, sono presenti i grafici di confronto generati tramite la piattaforma Weights & Biases, che mostrano l'andamento di varie metriche durante i test di training del modello RoBERTa. Questi grafici forniscono una visione dettagliata delle performance del modello nelle diverse epoche, evidenziando l'evoluzione delle metriche principali, come Precision, Recall, F1-score e Accuracy. Inoltre, sono stati aggiunti i classification report relativi a tali test, che non erano stati precedentemente inclusi nel corpo principale del lavoro. Questi report offrono un quadro completo delle performances, permettendo un confronto diretto tra le varie configurazioni e ottimizzazioni testate. L'inclusione di questi dati fornisce al lettore una visione più completa dell'intero processo di sviluppo e ottimizzazione, facilitando la comprensione dei risultati ottenuti e delle scelte fatte durante gli esperimenti.

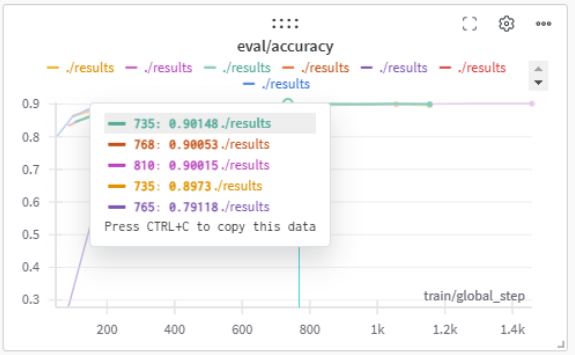


Figura A.1: wandb RoBERTa: confronto accuracy

Figura A.2: wandb RoBERTa: confronto recall

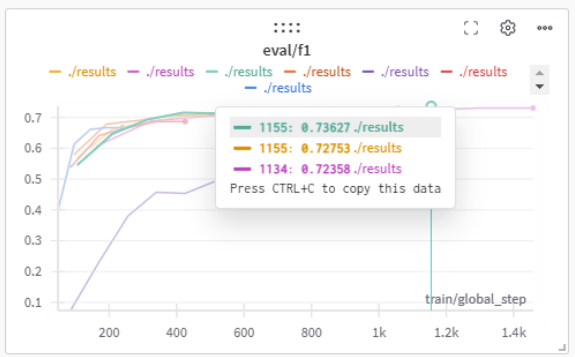


Figura A.3: wandb RoBERTa: confronto F1 score

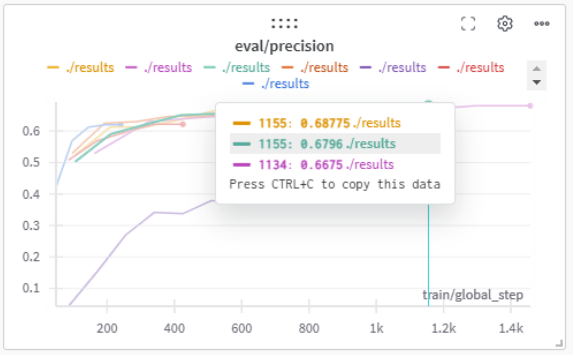


Figura A.4: wandb RoBERTa: confronto precision

Tabella A.1: RoBERTa test2: Class Report

Class	Pr	Rec	F1	Supp
B-DocumentReference	1.00	0.12	0.22	8
B-Location	0.76	0.81	0.79	155
B-MilitaryPlatform	0.41	0.44	0.42	16
B-Money	0.00	0.00	0.00	4
B-Nationality	0.75	0.30	0.43	10
B-Organisation	0.73	0.82	0.77	284
B-Person	0.85	0.94	0.90	106
B-Quantity	0.56	0.93	0.70	30
B-Temporal	0.72	0.89	0.80	46
B-Weapon	0.41	0.53	0.46	17
I-DocumentReference	0.57	0.80	0.67	20
I-Location	0.76	0.86	0.81	191
I-MilitaryPlatform	0.69	0.69	0.69	26
I-Money	0.00	0.00	0.00	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.77	0.84	0.80	411
I-Person	0.90	0.87	0.88	150
I-Quantity	0.77	0.53	0.63	32
I-Temporal	0.75	0.93	0.83	55
I-Weapon	0.40	0.43	0.41	14
O	0.96	0.92	0.94	3672
Accuracy			0.89	5258
Macro Avg	0.61	0.60	0.58	5258
Weighted Avg	0.90	0.89	0.89	5258

Tabella A.2: RoBERTa test3: Class Report

Class	Pr	Rec	F1	Supp
B-DocumentReference	0.15	1.00	0.26	8
B-Location	0.58	0.87	0.70	155
B-MilitaryPlatform	0.26	0.56	0.36	16
B-Money	0.17	1.00	0.30	4
B-Nationality	0.26	0.60	0.36	10
B-Organisation	0.62	0.84	0.71	284
B-Person	0.78	0.95	0.86	106
B-Quantity	0.51	0.97	0.67	30
B-Temporal	0.43	0.91	0.59	46
B-Weapon	0.16	0.88	0.28	17
I-DocumentReference	0.37	0.95	0.54	20
I-Location	0.53	0.93	0.68	191
I-MilitaryPlatform	0.53	0.88	0.67	26
I-Money	0.35	0.78	0.48	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.64	0.82	0.72	411
I-Person	0.78	0.88	0.82	150
I-Quantity	0.71	1.00	0.83	32
I-Temporal	0.56	0.93	0.70	55
I-Weapon	0.27	0.93	0.41	14
O	0.99	0.76	0.86	3672
Accuracy			0.79	5258
Macro Avg	0.46	0.83	0.56	5258
Weighted Avg	0.88	0.79	0.81	5258

Tabella A.3: RoBERTa test4: Class Report

Class	Pr	Rec	F1	Supp
B-DocumentReference	0.75	0.75	0.75	8
B-Location	0.75	0.83	0.79	155
B-MilitaryPlatform	0.36	0.31	0.33	16
B-Money	0.75	0.75	0.75	4
B-Nationality	0.86	0.60	0.71	10
B-Organisation	0.78	0.81	0.80	284
B-Person	0.89	0.92	0.90	106
B-Quantity	0.69	0.90	0.78	30
B-Temporal	0.75	0.91	0.82	46
B-Weapon	0.47	0.53	0.50	17
I-DocumentReference	0.73	0.80	0.76	20
I-Location	0.73	0.90	0.80	191
I-MilitaryPlatform	0.74	0.65	0.69	26
I-Money	1.00	0.78	0.88	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.80	0.84	0.82	411
I-Person	0.88	0.85	0.86	150
I-Quantity	0.97	0.97	0.97	32
I-Temporal	0.81	0.91	0.85	55
I-Weapon	0.50	0.57	0.53	14
O	0.96	0.93	0.94	3672
Accuracy			0.90	5258
Macro Avg	0.72	0.74	0.73	5258
Weighted Avg	0.90	0.90	0.90	5258

Tabella A.4: RoBERTa test6: Class Report

Class	Pr	Rec	F1	Supp
B-DocumentReference	1.00	0.88	0.93	8
B-Location	0.74	0.86	0.80	155
B-MilitaryPlatform	0.44	0.44	0.44	16
B-Money	0.75	0.75	0.75	4
B-Nationality	0.86	0.60	0.71	10
B-Organisation	0.78	0.80	0.79	284
B-Person	0.89	0.93	0.91	106
B-Quantity	0.64	0.90	0.75	30
B-Temporal	0.77	0.93	0.84	46
B-Weapon	0.42	0.47	0.44	17
I-DocumentReference	0.74	0.85	0.79	20
I-Location	0.76	0.87	0.81	191
I-MilitaryPlatform	0.80	0.62	0.70	26
I-Money	1.00	0.78	0.88	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.79	0.86	0.82	411
I-Person	0.88	0.91	0.89	150
I-Quantity	0.96	0.75	0.84	32
I-Temporal	0.83	0.95	0.88	55
I-Weapon	0.29	0.29	0.29	14
O	0.96	0.93	0.94	3672
Accuracy			0.90	5258
Macro Avg	0.73	0.73	0.72	5258
Weighted Avg	0.91	0.90	0.90	5258

Tabella A.5: RoBERTa test6: Class Report

Class	**Pr**	**Rec**	**F1**	**Supp**
B-DocumentReference	0.86	0.75	0.80	8
B-Location	0.76	0.83	0.79	155
B-MilitaryPlatform	0.47	0.44	0.45	16
B-Money	0.75	0.75	0.75	4
B-Nationality	0.88	0.70	0.78	10
B-Organisation	0.76	0.79	0.77	284
B-Person	0.91	0.93	0.92	106
B-Quantity	0.69	0.90	0.78	30
B-Temporal	0.72	0.93	0.81	46
B-Weapon	0.44	0.41	0.42	17
I-DocumentReference	0.73	0.80	0.76	20
I-Location	0.75	0.87	0.81	191
I-MilitaryPlatform	0.78	0.54	0.64	26
I-Money	1.00	0.78	0.88	9
I-Nationality	0.00	0.00	0.00	2
I-Organisation	0.79	0.84	0.81	411
I-Person	0.90	0.88	0.89	150
I-Quantity	1.00	0.72	0.84	32
I-Temporal	0.81	0.93	0.86	55
I-Weapon	0.29	0.29	0.29	14
O	0.95	0.93	0.94	3672
Accuracy			0.90	5258
Macro Avg	0.72	0.71	0.71	5258
Weighted Avg	0.90	0.90	0.90	5258

Bibliografia

[ZM24] Antonini Antonio Zazzarini Micol, Fiorani Andrea. Safeedgener. `https://github.com/MicolZazzarini/SafeEdgeNER`, 2024. Repository GitHub del progetto.