

Università degli studi di Parma

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA DEI SISTEMI INFORMATIVI

Realizzazione di un servizio per la deidentificazione automatizzata di documenti sanitari



**UNIVERSITÀ
DI PARMA**

Relatore:

Prof. Andrea Prati

Correlatore:

Vieri Emiliani

Candidato:
Riccardo Fava

Anno Accademico 2019-2020

*Dedico questo elaborato a tutte le persone che in questi anni mi hanno sostenuto,
supportato e aiutato a raggiungere questo importante traguardo...*

«Qualunque tecnologia sufficientemente avanzata è indistinguibile dalla magia.»
Arthur C. Clarke, autore del ciclo “Odissea nello spazio”

*«I computer sono incredibilmente veloci, accurati e stupidi.
Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.
L'insieme dei due costituisce una forza incalcolabile.»*
Albert Einstein

Ringraziamenti

Sommario

Questo elaborato si pone come obiettivo la realizzazione di un servizio per la **deidentificazione** automatizzata della documentazione clinica digitale, attraverso particolari tecniche di Machine Learning.

L'accelerazione che ha avuto il percorso di digitalizzazione dei servizi sanitari negli ultimi anni ha favorito un considerevole aumento della disponibilità di **documenti clinici** in formato elettronico, quali cartelle mediche, referti specialistici, verbali di pronto soccorso e lettere di dimissione ospedaliera.

Attraverso tecnologie di intelligenza artificiale sono stati sviluppati sistemi in grado di estrarre informazioni da una grande mole di dati sanitari distribuiti. Questi sistemi, tramite l'individuazione di correlazioni nascoste tra i dati, permettono il miglioramento di prevenzione, diagnosi, cura delle patologie, indagini epidemiologiche e strategie sanitarie.

L'analisi dell'ingente patrimonio informativo sanitario è però minata da due principali ostacoli: la maggior parte dei dati sanitari è prodotta in forma non strutturata, non consentendo quindi un'agevole estrazione di informazioni significative, inoltre i dati personali dei pazienti non possono essere utilizzati per fini statistici né resi pubblici per motivi connessi alla **Privacy**.

La grande quantità di informazioni sanitarie disponibili e il progressivo avanzamento delle tecnologie di analisi dei dati hanno dunque richiesto ai legislatori di tutto il mondo uno sforzo nella regolamentazione delle modalità di protezione dei dati personali.

Il progetto quindi verte sul privare i documenti dei **dati identificativi**, ovvero deidentificare le informazioni sensibili all'interno di un testo. Deidentificare significa eliminare la correlazione tra i dati personali e la persona fisica interessata, rendendo impossibile l'identificazione della stessa.

Lo sviluppo e la realizzazione di questo servizio potrebbe permettere un libero utilizzo della grande quantità di dati che vengono prodotti giornalmente dalle strutture sanitarie, il tutto rispettando le norme relative alla Privacy. Si farebbe quindi un enorme passo avanti dal punto di vista della ricerca in campo medico. Tutto questo porterebbe a migliorare significativamente la prevenzione, la diagnosi e la cura delle patologie e quindi alla concreta possibilità di **salvare vite umane**.

Indice

Ringraziamenti	3
Sommario	4
1 Introduzione	7
1.1 La necessità di questo servizio	8
1.2 Principi normativi sulla Privacy dei dati sanitari	9
1.3 Tecniche di anonimizzazione	11
1.3.1 Anonimizzazione	12
1.3.2 Pseudonimizzazione	13
2 Stato dell' arte	15
2.1 Machine Learning	15
2.2 Reti Neurali	17
2.3 Natural language Processing	18
2.4 La rappresentazione delle parole	21
2.4.1 One-hot vectors	21
2.4.2 Word embeddings	22
2.5 Algoritmi di named entity recognition	25
2.5.1 Anonimizzazione in documenti di casi legali francesi	26
3 Definizione del problema e approccio utilizzato	27
3.1 Criteri di annotazione utilizzati	28
3.2 Creazione di un dataset	30
3.2.1 Prodigy	30
4 Addestramento dei modelli	34
4.1 Modello baseline sviluppato con SpaCy	34
4.1.1 Spacy	34
4.1.2 Architettura della rete neurale	36
4.1.3 Addestramento del modello e risultati ottenuti	40
4.2 Modello intermedio sviluppato con SpaCy e FastText	42
4.2.1 FastText	42
4.2.2 Addestramento del modello e risultati ottenuti	43
4.3 Modello finale sviluppato con Flair e FastText	45
4.3.1 Flair	45
4.3.2 Architettura della rete neurale	46
4.3.3 Addestramento del modello e risultati ottenuti	49

5 Analisi dei risultati e confronto tra i modelli	53
5.1 Confronto tra i modelli	53
5.2 Match e Overlap	54
5.3 Confronto con il progetto della corte suprema francese	55
6 Visualizzazione predizioni del modello	58
6.1 Custom Recipe	58
7 Next steps e conclusioni	63
7.1 Next steps	63
7.2 Conclusioni	63
Bibliografia	64

Capitolo 1

Introduzione

Negli ultimi due anni le informazioni digitali di tutto il mondo sono più che raddoppiate e questa tendenza è destinata ad aumentare in modo esponenziale generando enormi moli di dati elettronici: i Big Data. La medicina è uno dei principali protagonisti di questa enorme crescita a causa della digitalizzazione di una grandissima quantità di documentazione clinica.

I documenti che gli organismi sanitari producono quotidianamente contengono una straordinaria quantità di informazioni, preziosissime in primo luogo per l'assistenza sanitaria, ma anche per la ricerca in campo medico.

Attraverso le tecniche di Machine learning i computer sono in grado di imparare dai dati, senza essere stati esplicitamente programmati per questo, generando modelli predittivi utili in moltissimi ambiti.

L'utilizzo dei Big Data necessariamente pone dei problemi per il rispetto della privacy. Gli individui vivono liberamente la propria vita nell'ipotesi che alcune informazioni personali non siano conosciute da chi non è autorizzato. Il prezzo dell'innovazione non può essere l'erosione del diritto fondamentale alla privacy, il quale viene assicurato da numerose leggi presenti a livello europeo e mondiale. Il binomio Privacy - Sanità da sempre presenta non poche difficoltà, sia per la rilevanza dei principi da tutelare, tutti di rango costituzionale, sia per l'approccio non sempre agevole degli operatori sanitari alle tematiche proprie della protezione dei dati personali; a questo si è poi sommato negli ultimi anni l'avvento della sanità digitale, che ha ulteriormente complicato la situazione.

Ecco quindi perché risultano essere fondamentali tecnologie in grado di deidentificare i documenti sanitari in modo tale da renderli interamente anonimi. Il seguente elaborato presenta uno strumento tramite il quale si possono utilizzare i dati sanitari per scopi di ricerca, rispettando però la Privacy e l'anonimato delle persone che li mettono a disposizione. Tutto questo nella prospettiva che l'apprendimento automatico non sia una bacchetta magica che può trasformare i dati in oro, ma diventi uno strumento prezioso, sempre più necessario per la medicina ed il sistema sanitario moderno, la cui complessità oggi supera la capacità della mente umana.

1.1 La necessità di questo servizio

“La ricerca scientifica è il motore per il progresso dell’umanità”. Da questa frase si può dedurre l’importanza fondamentale della ricerca, per quanto riguarda la nostra vita e per il futuro del pianeta in cui viviamo. In particolare la ricerca in ambito medico ci ha permesso di migliorare sempre di più quelle che sono le tecniche di diagnosi, cura e prevenzione delle malattie.

Il periodo che purtroppo stiamo vivendo, caratterizzato dalla pandemia dovuta al virus COVID-19, è un esempio eclatante di quanto siano importanti gli studi scientifici per la formulazione di strategie e cure utili per combattere questa terribile malattia. Strumento fondamentale in questo caso, come in molti altri, risultano essere il machine learning e l’intelligenza artificiale. In questo caso, attraverso particolari algoritmi, è possibile infatti tentare di prevedere le mosse del “nemico”, in questo caso il virus, responsabile dell’epidemia, e di comprendere aspetti del fenomeno che altrimenti ci sarebbero sconosciuti o che verrebbero poi scoperti sul campo in modo tardivo. Diventa dunque quasi scontato dire che in questo scenario i dati che descrivono dove ci troviamo e come ci spostiamo sono di fondamentale importanza per tentare di prevedere il modo in cui un virus si diffonderà tra la popolazione.

Da questo esempio contemporaneo è possibile evincere l’importanza del dato, che diventa vero e proprio carburante per gli algoritmi di machine learning.

Nel contesto dell’elaborazione delle informazioni sanitarie, un aspetto fondamentale da tenere in conto riguarda la tutela del diritto di privacy di ogni cittadino/paziente. Molti organismi di pubblici e privati si scontrano molto spesso con gli enti sanitari riguardo queste delicate tematiche: da una parte chi si occupa di ricerca tenta in ogni modo di ottenere la grande quantità di dati a disposizione dell’azienda ospedaliera; dall’altra gli organismi sanitari, a rispetto delle leggi sul trattamento dei dati personali, risultano essere molto restii nel fornire il patrimonio di cui sono in possesso. Ci si trova quindi in un’inevitabile situazione di stallo.

Una soluzione possibile che risolve quindi questo problema, risulta essere utilizzare tecniche di anonimizzazione a monte del trattamento dei dati.

Giungiamo quindi alla necessità di sviluppare un metodo per deidentificare i documenti sanitari in un modo semplice, veloce ed efficiente.

1.2 Principi normativi sulla Privacy dei dati sanitari

Privacy è un termine inglese che viene tradotto in italiano come riservatezza. Con questo termine si intende il diritto di ogni persona di mantenere i propri dati riservati o gestiti solo da chi ne ha ottenuto l'autorizzazione. Il concetto di privacy assume connotazioni diverse, specie se si tratta di ambito giornalistico e mediatico. In sanità la privacy comprende la protezione dei dati personali del paziente riguardanti il suo stato di salute e le corrispondenti modalità di trattamento. Ogni cittadino che accede ad una struttura sanitaria per visite, esami o ricoveri necessita infatti che gli venga garantita l'assoluta riservatezza, nel rispetto dei suoi diritti fondamentali e della sua dignità.

Le leggi sulla privacy sono contenute nel codice sulla privacy approvato dal decreto legislativo 196/2003 [1], che al suo interno contiene tutte le disposizioni in materia di trattamento dei dati personali, già in vigore nelle leggi precedenti.

Il codice garantisce che il trattamento dei dati personali si svolga nel rispetto dei diritti e della libertà fondamentali, nonché della dignità dell'interessato, con particolare riferimento alla riservatezza, all'identità personale e al diritto alla protezione dei dati. Inoltre, obbliga al rispetto del principio di necessità nel trattamento dei dati, riducendo così l'utilizzo a finalità specifiche e ben stabilite. In particolare, ad esempio, i presupposti di liceità del trattamento dei dati idonei a rivelare lo stato di salute da parte degli esercenti le professioni sanitarie e degli organismi sanitari pubblici vengono definiti dall'art. 76 del codice. Tale trattamento è effettuato con il consenso dell'interessato e anche senza l'autorizzazione del Garante, se riguarda dati e operazioni indispensabili per perseguire una finalità di tutela della salute o dell'incolumità fisica dell'interessato, ovvero anche senza il consenso dell'interessato e previa autorizzazione del Garante, se la finalità riguarda un terzo o la collettività.

Come sappiamo, poi, è sopraggiunto il Regolamento UE n. 2016/679 (GDPR)[2] del Parlamento Europeo del 27 aprile 2016 relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali. Il GDPR (General Data Protection Regulation), ha avuto lo scopo di uniformare le normative di riferimento sulla privacy tra i Paesi membri.

Questa normativa europea non prevede riferimenti specifici per il trattamento dei dati personali effettuato in ambito sanitario. Naturalmente però, la materia è oggetto di particolare attenzione da parte del legislatore comunitario, il quale nel "Considerando 35", chiarisce che: *"nei dati personali relativi alla salute dovrebbero rientrare tutti i dati riguardanti lo stato di salute dell'interessato che rivelino informazioni connesse allo stato di salute fisica o mentale passata, presente o futura dello stesso. Questi comprendono informazioni sulla persona fisica raccolte nel corso della sua registrazione al fine di ricevere servizi di assistenza sanitaria o della relativa prestazione di cui alla direttiva 2011/24/UE del Parlamento europeo e del Consiglio; un numero, un simbolo o un elemento specifico attribuito a una*

persona fisica per identificarla in modo univoco a fini sanitari; le informazioni risultanti da esami e controlli effettuati su una parte del corpo o una sostanza organica, compresi i dati genetici e i campioni biologici; e qualsiasi informazione riguardante, ad esempio, una malattia, una disabilità, il rischio di malattie, l'anamnesi medica, i trattamenti clinici o lo stato fisiologico o biomedico dell'interessato, indipendentemente dalla fonte, quale, ad esempio, un medico o altro operatore sanitario, un ospedale, un dispositivo medico o un test diagnostico in vitro”.

Una definizione estremamente completa e dettagliata sintetizzata in qualche modo dall'art. 4, n. 15 del GDPR che per dati relativi alla salute intende: i dati personali attinenti alla salute fisica o mentale di una persona fisica, compresa la prestazione di servizi di assistenza sanitaria, che rivelano informazioni relative al suo stato di salute.

Sul tema della tutela della privacy anche gli Stati Uniti hanno prestato un grande interesse. Già dal 1996 è infatti in vigore l'HIPAA (Health Insurance Portability and Accountability Act)[\[3\]](#), la legge federale che regola il settore delle assicurazioni in campo sanitario. Questa legge fornisce importanti indicazioni sulla protezione dei dati sanitari dei pazienti identificando 18 tipi di informazioni sanitarie protette (PHI, Protected Health Information), che includono nomi propri, identificativi geografici, date correlate all'interessato, recapiti e così via. Tali informazioni non devono essere considerate per scopi differenti da quelli di cura.

Le norme stabilite dall'HIPAA includono sia le norme relative alla riservatezza dei pazienti (Privacy Rule), sia quelle relative alla protezione delle informazioni (Security Rule). La Privacy Rule conferisce ai singoli individui il diritto di controllare le proprie informazioni sanitarie e stabilisce norme e limiti in merito al loro uso e alla loro divulgazione. Si applica a tutte le informazioni sanitarie protette (PHI) dei pazienti, trasmesse per iscritto, in via telematica oppure oralmente.

La Security Rule invece è una legge federale che protegge le informazioni sanitarie in formato elettronico (ePHI).

Nella seguente immagine possiamo vedere quali sono i 18 tipi di informazioni sanitarie protette identificate dalla legge HIPAA.

(2)(i) The following identifiers of the individual or of relatives, employers, or household members of the individual, are removed:																	
<p>(A) Names</p> <p>(B) All geographic subdivisions smaller than a state, including street address, city, county, precinct, ZIP code, and their equivalent geocodes, except for the initial three digits of the ZIP code if, according to the current publicly available data from the Bureau of the Census:</p> <ul style="list-style-type: none"> (1) The geographic unit formed by combining all ZIP codes with the same three initial digits contains more than 20,000 people; and (2) The initial three digits of a ZIP code for all such geographic units containing 20,000 or fewer people is changed to 000 <p>(C) All elements of dates (except year) for dates that are directly related to an individual, including birth date, admission date, discharge date, death date, and all ages over 89 and all elements of dates (including year) indicative of such age, except that such ages and elements may be aggregated into a single category of age 90 or older</p>																	
<table border="1"> <tr> <td>(D) Telephone numbers</td><td>(L) Vehicle identifiers and serial numbers, including license plate numbers</td></tr> <tr> <td>(E) Fax numbers</td><td>(M) Device identifiers and serial numbers</td></tr> <tr> <td>(F) Email addresses</td><td>(N) Web Universal Resource Locators (URLs)</td></tr> <tr> <td>(G) Social security numbers</td><td>(O) Internet Protocol (IP) addresses</td></tr> <tr> <td>(H) Medical record numbers</td><td>(P) Biometric identifiers, including finger and voice prints</td></tr> <tr> <td>(I) Health plan beneficiary numbers</td><td>(Q) Full-face photographs and any comparable images</td></tr> <tr> <td>(J) Account numbers</td><td>(R) Any other unique identifying number, characteristic, or code, except as permitted by paragraph (c) of this section; and</td></tr> <tr> <td>(K) Certificate/license numbers</td><td></td></tr> </table>		(D) Telephone numbers	(L) Vehicle identifiers and serial numbers, including license plate numbers	(E) Fax numbers	(M) Device identifiers and serial numbers	(F) Email addresses	(N) Web Universal Resource Locators (URLs)	(G) Social security numbers	(O) Internet Protocol (IP) addresses	(H) Medical record numbers	(P) Biometric identifiers, including finger and voice prints	(I) Health plan beneficiary numbers	(Q) Full-face photographs and any comparable images	(J) Account numbers	(R) Any other unique identifying number, characteristic, or code, except as permitted by paragraph (c) of this section; and	(K) Certificate/license numbers	
(D) Telephone numbers	(L) Vehicle identifiers and serial numbers, including license plate numbers																
(E) Fax numbers	(M) Device identifiers and serial numbers																
(F) Email addresses	(N) Web Universal Resource Locators (URLs)																
(G) Social security numbers	(O) Internet Protocol (IP) addresses																
(H) Medical record numbers	(P) Biometric identifiers, including finger and voice prints																
(I) Health plan beneficiary numbers	(Q) Full-face photographs and any comparable images																
(J) Account numbers	(R) Any other unique identifying number, characteristic, or code, except as permitted by paragraph (c) of this section; and																
(K) Certificate/license numbers																	
<p>(ii) The covered entity does not have actual knowledge that the information could be used alone or in combination with other information to identify an individual who is a subject of the information.</p>																	

Figura 1.1: Informazioni sanitarie protette identificate dalla legge HIPAA.

1.3 Tecniche di anonimizzazione

Il GDPR si limita a definire alcuni punti cardine per garantire la sicurezza all'interno del processo di gestione dei dati personali; viene però lasciata ai titolari e ai responsabili del trattamento una certa libertà di scelta in merito agli accorgimenti tecnici specifici da impiegare. Questa libertà di azione ha creato non poca confusione nella definizione dei limiti entro cui operare ed entro cui le disposizioni del GDPR possono essere applicate. I dubbi nascono dalla compresenza di diverse soluzioni tecniche, molto spesso confuse tra loro, che quindi alimentano il disordine interpretativo in materia: l'anonimizzazione e la pseudonimizzazione sono le più importanti. Queste soluzioni definiscono il contesto entro cui districarsi per garantire un'adeguata e concreta protezione dei dati personali.

1.3.1 Anonimizzazione

Per anonimizzazione^[4] si intende una tecnica che viene applicata ai dati personali in modo tale che le persone fisiche interessate non possano più essere identificate in nessun modo: l'obiettivo è ottenere una deidentificazione irreversibile. Nel momento in cui i dati personali riferiti ad un individuo sono stati adeguatamente anonimizzati, dovrebbe essere impossibile poter invertire il processo, che risulta quindi irreversibile. Requisito fondamentale è che i dati personali siano stati inizialmente raccolti, trattati e conservati in conformità alla normativa vigente, con riferimento ai principi applicabili al trattamento e alla liceità dello stesso, ai sensi degli articoli 5 e 6 del GDPR.

Se l'anonimizzazione è andata a buon fine, i dati oggetto dell'operazione non sono più classificati come dati personali e quindi non rientrano nella dimensione applicativa del GDPR. A sostegno di ciò, il “Considerando 26” del GDPR specifica che: *“I principi di protezione dei dati non dovrebbero pertanto applicarsi a informazioni anonime, vale a dire informazioni che non si riferiscono a una persona fisica identificata o identificabile o a dati personali resi sufficientemente anonimi da impedire o da non consentire più l'identificazione dell'interessato”*. Da ricordare, inoltre, che i dati anonimizzati sono compresi fra gli esempi specifici di “dati non personali”, così come definito nel “Considerando 9” del “Regolamento UE 2018/1807 relativo alla libera circolazione dei dati non personali nell’Unione Europea”.

L'anonimizzazione è uno strumento prezioso che consente la condivisione di set di dati, garantendo sia la privacy delle persone fisiche che la possibilità di sfruttare queste informazioni per analisi e ricerche statistiche. L'anonimizzazione si può realizzare tramite la rimozione, la sostituzione, la distorsione, la generalizzazione o l'aggregazione degli identificatori diretti, come il nome completo o altre caratteristiche rilevanti della persona fisica.

Uno dei metodi più comuni per anonimizzare dei dati comporta l'eliminazione degli identificatori diretti. Questa singola operazione spesso però non è però sufficiente a garantire che l'identificazione della persona interessata non sia più possibile. Per questo si consiglia di non utilizzare un'unica tecnica di anonimizzazione, ma una combinazione di esse. All'eliminazione degli identificatori diretti si può aggiungere a titolo esemplificativo la tecnica della generalizzazione, la quale comporta la riduzione del grado di dettaglio di una determinata variabile. Per esempio, le date di nascita di singole persone fisiche possono essere generalizzate per mese o anno, producendo una riduzione del grado di identificabilità. Quindi, eliminare i nomi completi degli individui, mantenendo solo l'anno di nascita degli stessi, permette di deidentificare in modo irreversibile le persone fisiche, potendo comunque effettuare analisi statistiche sul campione di dati (Figura 1.2).

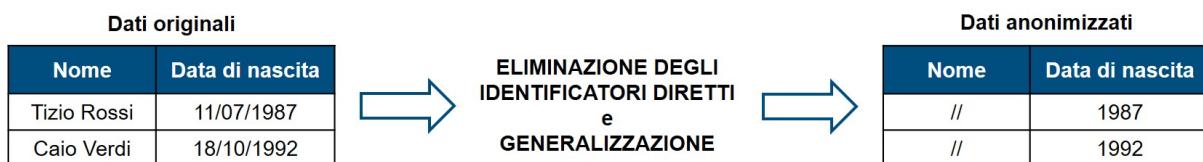


Figura 1.2: Esempio di tecniche di anonimizzazione

1.3.2 Pseudonimizzazione

La pseudonimizzazione è definita, nel disposto dell’art. 4 Punto 5 del GDPR, come “*il trattamento dei dati personali in modo tale che i dati personali non possano più essere attribuiti a un interessato specifico senza l’utilizzo di informazioni aggiuntive [...]*”. In concreto, questa operazione sostituisce alcuni identificatori con pseudonimi cioè dati realistici, ma non veritieri. I dati originali, vengono conservati in un database separato costituito da una tabella delle corrispondenze tra dati originali e gli pseudonimi utilizzati. Tutto ciò permette di re-identificare le persone fisiche, in quanto il titolare, o il responsabile del trattamento, possiede le “informazioni aggiuntive” per poter risalire all’identità degli interessati: si parla perciò di un processo reversibile.

I dati pseudonimizzati non possono essere attribuiti ad un individuo senza utilizzare le predette “informazioni aggiuntive”, che in questo caso sono rappresentate dalla tabella delle corrispondenze tra pseudonimi e dati originali. Importante ricordare che le “informazioni aggiuntive” devono essere conservate in un database separato e adeguatamente protetto. Infatti, in questo modo anche se il set di dati pseudonimizzati venisse compromesso, non sarà comunque possibile risalire ai dati originali.

La pseudonimizzazione riduce il rischio di identificazione diretta degli individui, riducendo la correlabilità di un insieme di dati all’identità originaria di una persona interessata, ma non produce dati anonimi. Quindi, i dati pseudonimizzati sono dati personali e rientrano di conseguenza nella disciplina del GDPR. Infatti, il “Considerando 26” del GDPR chiarisce eventuali dubbi dichiarando che “*i dati personali sottoposti a pseudonimizzazione, i quali potrebbero essere attribuiti a una persona fisica mediante l’utilizzo di ulteriori informazioni, dovrebbero essere considerati informazioni su una persona fisica identificabile*”.

Chiarito che la pseudonimizzazione non è un metodo di anonimizzazione, è opportuno ricordare che l’art. 32 comma 1 lett. a) del GDPR cita questa misura tecnica, definendola appropriata “*per garantire un livello di sicurezza adeguato al rischio*”.

La pseudonimizzazione è uno strumento fondamentale per proteggere i dati personali, ma perché viene scelta questa tecnica rispetto all’anonimizzazione? Di fatto, il principale vantaggio offerto dalla pseudonimizzazione è proprio quello di poter risalire ai dati originali avendo comunque la possibilità di divulgare i dati

pseudonimizzati senza rischio di re-identificazione. Questa possibilità ha molteplici risvolti applicativi.

Capitolo 2

Stato dell' arte

Illustriamo ora lo stato dell'arte delle tecniche e delle tecnologie che sono state utilizzate per la realizzazione del servizio di deidentificazione.

2.1 Machine Learning

La definizione più citata di machine learning o apprendimento automatico è quella fornita da Tom M. Mitchell [5]: *“Si dice che un programma apprende dall'esperienza E con riferimento ad alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E”*.

Il Machine Learning, è una branca dell'intelligenza artificiale che raccoglie metodi sviluppati negli ultimi decenni del XX secolo in varie comunità scientifiche. Definire in maniera semplice le caratteristiche e le applicazioni del machine learning non è sempre possibile, visto che questo ramo è molto vasto e prevede differenti modalità, tecniche e strumenti per essere realizzato. Si può tuttavia definire il machine learning come una serie di differenti meccanismi che permettono a una macchina intelligente di migliorare le proprie capacità e prestazioni nel tempo. La macchina, quindi, sarà in grado di imparare a svolgere determinati compiti migliorando, tramite l'esperienza, le proprie capacità, le proprie risposte e funzioni. Alla base dell'apprendimento automatico ci sono una serie di differenti algoritmi che, partendo da un set di dati, sapranno prendere una specifica decisione piuttosto che un'altra o effettuare azioni apprese nel tempo.

Le modalità di utilizzo dell'apprendimento automatico vengono solitamente classificate in tre categorie, o paradigmi:

- *Apprendimento supervisionato*: si forniscono degli esempi, che consistono in una coppia costituita da un vettore di input ed un valore di output desiderato. L'obiettivo dell'algoritmo è quello di estrarre una regola generale che associa il valore di input al valore di output corrispondente. L'idea di base è quella di trovare un insieme di pesi che siano in grado di predire correttamente l'output desiderato dato qualsiasi valore di input, generalizzando da dati di apprendimento a situazioni differenti. Si possono avere due differenti

situazioni: la classificazione e la regressione. Attraverso la classificazione si cerca di classificare i dati in un gruppo di determinate categorie. Attraverso la regressione invece dato un insieme di variabili vogliamo andar a prevedere il valore futuro della nostra variabile oggetto dell'analisi.

- *Apprendimento non supervisionato*: abbiamo unicamente i dati di input senza alcun output desiderato. Lo scopo del calcolatore è quindi quello di identificare dei pattern di correlazione all'interno degli input. Gli algoritmi di apprendimento non supervisionato possono essere suddivisi in tre differenti situazioni: clustering (scoprire raggruppamenti nei dati), analisi delle associazioni (scoprire le regole che meglio riescono a descrivere la maggior porzione di dati possibile) o quantile estimation.
- *Apprendimento per rinforzo*: prevede l'interazione del calcolatore con un ambiente dinamico nel quale si cerca di raggiungere un obiettivo. L'obiettivo è classificare differenti situazioni al fine di intraprendere un'azione.

Ma quali sono le modalità di addestramento e creazione di un modello di machine learning in grado di imparare dai dati che gli vengono forniti?

Innanzitutto bisogna scegliere quale tra i paradigmi presentati in precedenza utilizzare. Nella maggior parte dei casi, e come vedremo anche nel nostro progetto, l'approccio supervisionato è quello che risulta essere il più utilizzato. Illustriamo quindi le modalità di addestramento di un modello secondo questo paradigma.

1. La prima cosa da fare è creare un dataset, ovvero un file all'interno del quale inserire i dati in un formato comprensibile dagli algoritmi di addestramento del modello.
2. Il dataset deve essere successivamente diviso in set di *Train* e set di *Test*. Il set di Train verrà utilizzato dall'algoritmo o dalla rete neurale per addestrare il modello di machine learning. Il set di Test invece verrà utilizzato per effettuare una valutazione delle prestazioni del modello una volta terminata la fase di training. Questa fase risulta essere molto importante in quanto il set di Test dovrà rimanere fuori dal processo di addestramento in modo tale da poter valutare successivamente il modello con dati che non ha mai visto.
3. Addestramento del modello fornendogli in input solamente il set di Train.
4. Una volta terminato il training del modello sarà possibile valutarne le prestazioni in termini di Precision, Recall e F-Score, attraverso l'utilizzo del Test set.

2.2 Reti Neurali

Una rete neurale artificiale (Artificial Neural Network) è un sistema di intelligenza artificiale ispirato alle reti neurali biologiche. Si tratta di un modello matematico di calcolo basato su percetroni, neuroni artificiali in grado di apprendere, ovvero accumulare esperienza. L'elemento costituente più rilevante delle reti neurali è l'implementazione di un approccio di connessioni dinamiche; la rete neurale è quindi in grado di modificare la propria struttura variando in risposta ai flussi di dati che determinano l'apprendimento.

Ciascuno dei nodi della rete rappresenta un'unità di calcolo adattiva, poiché il proprio output dipende da parametri modificabili. I neuroni artificiali, infatti, sono in grado di variare i propri parametri di calcolo sulla base dei dati di training.

Un perceptron, rappresentato di seguito, accetta un certo numero di input dagli altri neuroni artificiali e produce in output un segnale. Questo segnale viene ottenuto combinando l'input con numerosi pesi e successivamente facendolo elaborare da una specifica funzione matematica chiamata funzione di attivazione.

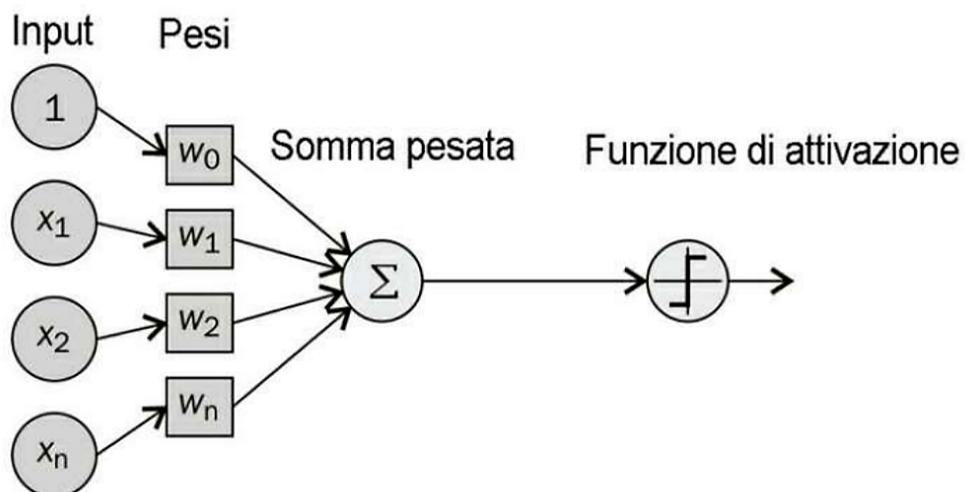


Figura 2.1: Struttura e meccanismo di elaborazione dei percetroni

Per creare una rete neurale dobbiamo connettere fra loro più perceptron. I collegamenti tra neuroni hanno un valore associato, detto peso sinaptico, il quale ha lo scopo di amplificare o ridurre l'importanza che un dato neurone ha all'interno della rete.

Una rete neurale è composta da:

- Un layer di ingresso, formato dai neuroni di input, attraverso i quali sono forniti di dati.
- Uno o più layer intermedi (hidden layers) che eseguono elaborazioni dei dati.
- Un layer di output, che fornisce il risultato

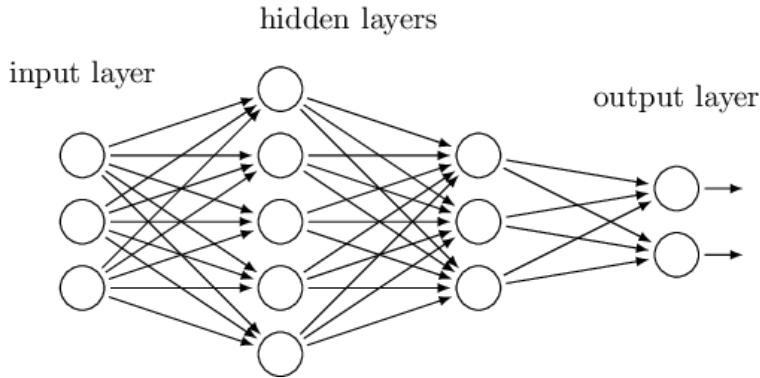


Figura 2.2: Struttura di una rete neurale artificiale

Mentre addestriamo il modello i pesi, inizialmente casuali, vengono aggiornati in modo da ottenere la migliore previsione possibile. Ma come è possibile ottenere un miglioramento progressivo del modello? Una soluzione ricorrente è quella della retropropagazione dell'errore, o backpropagation. Durante questo processo, la rete neurale calcola il gradiente dell'errore (ottenuto tramite una funzione di loss) sui dati di output, rispettivamente ai vari pesi dei layer della rete. Questo permette alla rete neurale di individuare quali nodi hanno più effetto sulla generazione dell'errore, e di conseguenza, regolarne i pesi per ridurre l'errore. La ricorsione di questo processo permette di creare un modello sempre più preciso e performante.

Quando le reti neurali crescono diventando molto profonde, entriamo nel campo dell'apprendimento profondo (Deep Learning). Il grande vantaggio delle reti neurali profonde (formate da molti livelli) è il fatto che sono in grado di approssimare quasi ogni funzione e, teoricamente, possono apprendere le combinazioni ottimali di caratteristiche e poi usarle per ottenere il massimo potere predittivo possibile.

2.3 Natural language Processing

L'elaborazione del linguaggio naturale (in inglese Natural Language Processing, NLP) è una branca, un campo di studi e di ricerca che si divide tra intelligenza artificiale e linguistica computazionale. Essa fa riferimento al processo di trattamento automatico mediante un calcolatore delle informazioni scritte o parlate in lingua naturale, ponendosi come vero e proprio modello di mediazione nell'interazione uomo-macchina.

La complessità che sta alla base di questo processo, dovuta alle caratteristiche intrinseche di ambiguità del linguaggio umano, è affrontata suddividendo questo processo in fasi diverse: analisi lessicale, grammaticale, sintattica, semantica.

Nel 2011, per la prima volta un semplice algoritmo basato sul deep learning è stato applicato a differenti problemi di NLP, tra cui l'identificazione di entità e l'assegnazione di categorie morfologiche a parole, mostrando prestazioni sensibilmente migliori rispetto ad altri approcci rappresentativi dello stato dell'arte. Da allora, sono stati realizzati algoritmi sempre più complessi basati sul deep

learning per affrontare problemi di NLP ancora non risolti o trattati in passato ma con risultati non soddisfacenti.

Ai giorni nostri il NLP è in continua espansione grazie alle nuove tecnologie informatiche. Ricopre tuttora un ruolo fondamentale in tutti i campi in cui si ha a che fare con linguaggio scritto o parlato, come ad esempio l'Information Retrieval, gli assistenti vocali e la traduzione automatica.

La potenza sempre più elevata degli elaboratori ha facilitato molto l'elaborazione e la computazione di algoritmi in grado di processare enormi moli di dati. Ovviamente siamo ancora lontani da sistemi perfetti, in grado di lavorare al 100% delle possibilità e di garantire una conoscenza globale dell'ambiente con il quale si vanno ad interfacciare. Il Natural language processing però ha fatto enormi passi avanti, ed è entrato a far parte della nostra vita di tutti i giorni grazie alle numerose applicazioni che ne fanno uso.

In maggior dettaglio, in primo luogo, l'NLP fornisce soluzioni per analizzare la struttura sintattica del testo, associando alle singole parole le rispettive categorie morfologiche (ad es. nome, verbo, aggettivo), identificando entità e classificandole in categorie predefinite (ad es. persona, data, luogo), estraendo dipendenze sintattiche (ad es. soggetti e complementi) e relazioni semantiche. In secondo luogo, consente di comprendere la semantica del testo, identificando il significato delle parole, anche relativamente al contesto e alle modalità di utilizzo (ad es. ironia, sarcasmo, sentimento, umore), classificandolo in categorie predefinite (ad es. sport, geografia, medicina) o sintetizzandone il contenuto.

Tutte queste tipologie di analisi del linguaggio naturale si possono riassumere in una serie di operazioni di preprocessing dei dati attuabili su di un testo scritto:

- *Tokenizzazione*: Una stringa di caratteri, deve essere suddivisa in parole o, più precisamente, in token diversi. I token sono separati l'uno dall'altro da appositi caratteri, i separatori, che possono essere spazi bianchi, tabulazioni, terminatori di paragrafo o elementi di punteggiatura. Alcuni elementi di punteggiatura possono inoltre essere considerati come token come ad esempio punti interrogativi o esclamativi.

0	1	2	3	4	5	6	7	8	9	10
Apple	is	looking	at	buying	U.K.	startup	for	\$	1	billion

Figura 2.3: Esempio di tokenizzazione di una frase

- *Stemming*: Una volta riconosciuti i token, è importante trasformarli in una forma standard. Se la stessa parola è presente, ad esempio, tre volte in forme diverse (es. singolare/plurale, iniziale maiuscola/minuscola), è opportuno che le tre forme siano contate come tre istanze della stessa parola. Attraverso un processo chiamato lemmatizzazione è possibile ricondurre a un unico lemma

la coniugazione o la declinazione di parole differenti. Ad esempio, le parole andiamo o vado si possono ricondurre al lemma andare.

- *Part of Speech (POS) Tagging*: Dopo aver diviso il testo in token, si può quindi procedere con un’assegnazione del ruolo grammaticale (part of speech) ad essi. Si può quindi attribuire ad ogni token il ruolo di nome, verbo, aggettivo, avverbio, ecc...
- *Dependency Parsing*: Il parsing consiste nell’analisi delle relazioni tra i token all’interno una frase, e nell’assegnazione di un ruolo semantico. Si andrà per esempio ad identificare il soggetto e l’oggetto all’interno di ciascuna frase presente nel testo.

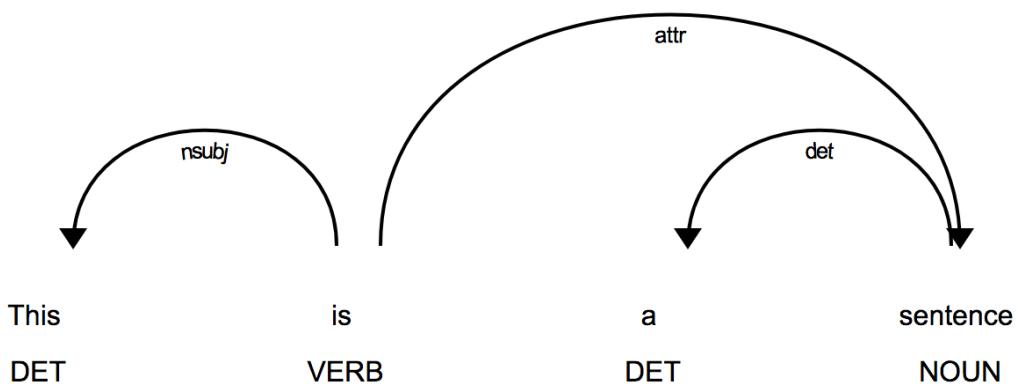


Figura 2.4: Esempio di part of speech e dependency parsing

- *Named Entity Recognition*: Attraverso questo tipo di analisi è possibile rintracciare all’interno del testo un determinato numero di entità ricorrenti. Ad esempio, si possono voler riconoscere nomi di luoghi, di aziende, espressioni temporali o altri tipi di informazioni che possono essere ricondotte a particolari categorie di entità.
Questo particolare tipo di analisi sarà, come vedremo in seguito, oggetto di particolare attenzione per la realizzazione del progetto.

2.4 La rappresentazione delle parole

Un problema fondamentale per tutte le applicazioni del NLP è quello di rappresentare le parole dei testi all’interno del calcolatore: abbiamo bisogno di rappresentare il testo scritto in modo tale da poterlo processare in modo automatico, in modo analogo a quanto avviene con le immagini e le onde acustiche, rappresentate da segnali analogici o digitali.

2.4.1 One-hot vectors

L'approccio più semplice per la rappresentazione delle parole è quello di utilizzare i cosiddetti “one-hot embeddings”, attraverso cioè vettori di grandi dimensioni (tante quante la dimensione del dizionario) con componenti tutte uguali a zero, tranne per quella relativa all'indice della parola rappresentata.

Sia S l'insieme senza ripetizioni di tutti i token del dataset. Ogni elemento $s \in S$ rappresenti una dimensione di un vettore di booleani. Ogni istanza del dataset è rappresentata con un vettore di booleani (1 se il token è presente, 0 altrimenti).

Esempio:

- dataset $D = \{"\text{oggi è mercoledì}", "\text{oggi non piove}", "\text{se piove prendo un ombrello}"\}$
- $S = [\text{'mercoledì}', \text{'non}', \text{'oggi'}, \text{'ombrello'}, \text{'piove'}, \text{'prendo'}, \text{'se'}, \text{'un'}, \text{'è'}]$

	mercoledì	non	oggi	ombrello	piove	prendo	se	un	è
oggi è mercoledì	1	0	1	0	0	0	0	0	1
oggi non piove	0	1	1	0	1	0	0	0	0
se piove prendo un ombrello	0	0	0	1	1	1	1	1	0

Figura 2.5: Esempio di rappresentazione di frasi tramite i one-hot vectors

Si tratta di una rappresentazione molto semplice e facile da costruire, ma presenta tutta una serie di svantaggi che la rendono non molto efficiente in un sistema reale: innanzitutto la dimensione dello spazio vettoriale generato dipende dalla dimensione del dizionario, che solitamente è molto grande. Inoltre questo tipo di rappresentazione non fornisce informazioni sulle relazioni semantiche tra le diverse parole, dato che si tratta di vettori ortogonali tra loro, e che quindi non permettono l'estrazione di un valore di similarità.

2.4.2 Word embeddings

Con il termine Word Embeddings viene indicato un insieme di strumenti, modelli del linguaggio e tecniche di apprendimento all'interno dell'area del Natural Language Processing che permettono di rappresentare parole e frasi di testi scritti attraverso vettori di numeri reali.

L'approccio è quello di provare a rappresentare le parole di un testo non più attraverso vettori di grandi dimensioni a componenti discrete, ma attraverso vettori

di dimensione inferiore, ma con componenti continue. E' questo il caso dei Word Embeddings, che sono quindi in grado di codificare le informazioni semantiche e sintattiche delle parole, dove l'informazione semantica ha a che fare con il significato, mentre quella sintattica ha a che fare con il ruolo all'interno della struttura del testo. A questo punto è possibile sviluppare modelli linguistici e statistici che permettono, a partire da un corpus di documenti iniziale, di addestrare tramite reti neurali, tecniche di Machine Learning e Deep Learning questi vettori.

La maggior parte di questi modelli cercano di ottimizzare una funzione di loss andando a minimizzare la differenza tra i valori della predizione e quelli reali.

In letteratura sono presenti vari approcci ai Word Embeddings, che possono essere classificati in base alla distribuzione dell'informazione delle parole:

- Modelli di word embeddings standard, all'interno dei quali viene definita in fase di training una corrispondenza biunivoca tra una parola e un vettore. Esempi di framework che lavorano con questo tipo di word embeddings sono Word2Vec[6][7], GolVe[8] e FastText[9].
- Modelli di embedding contestuali, che pongono l'attenzione sul contesto all'interno del quale appaiono le parole presenti nel testo. Questo tipo di embedding viene generato a runtime dal modello, considerando il contesto all'interno del quale si trova la parola. Esempi di framework che lavorano con questo tipo di word embeddings sono BERT[10] e Flair[11].

Nella seguente immagine è possibile vedere come parole aventi significato o contesto simile abbiano una rappresentazione vettoriale molto vicina all'interno dei word embeddings.

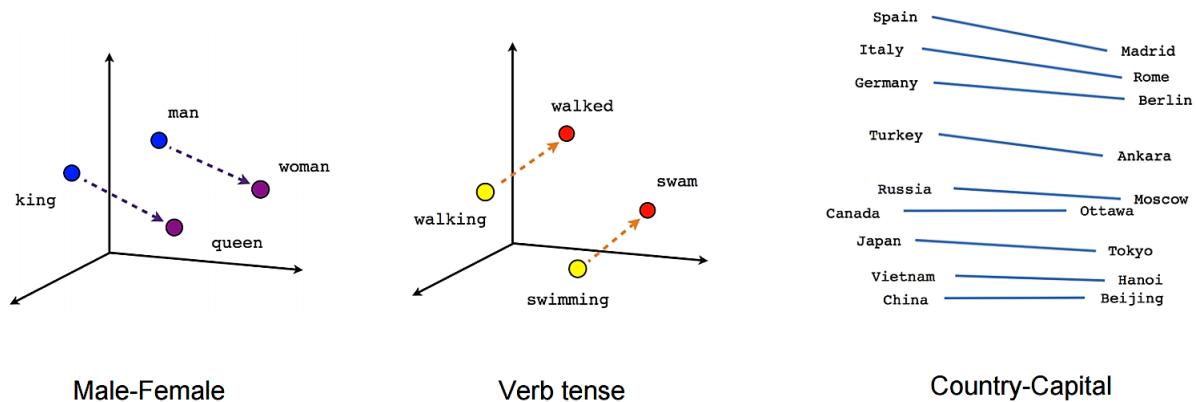


Figura 2.6: Esempio di rappresentazione di parole simili tramite i word-embeddings

Word2Vec

Uno dei modelli più famosi usati per l'apprendimento delle rappresentazioni vettoriali delle parole è Word2Vec (Mikolov, Chen, Corrado, & Dean, 2013)[6][7]. In particolare, nel loro articolo, gli autori propongono due architetture per l'addestramento di Word Embeddings di buona qualità da collezioni di documenti molto grandi, composti da miliardi di parole e con una particolare attenzione alla complessità computazionale. Tali modelli sono il Continuous Bag-of-Words (CBOW) e lo Skip-Gram (SG). L'addestramento viene effettuato cercando di ottimizzare una funzione obiettivo che altro non è che il calcolo della probabilità di predizione di una certa parola centrale o di un insieme di parole contesto all'interno di una finestra di dimensione 2D.

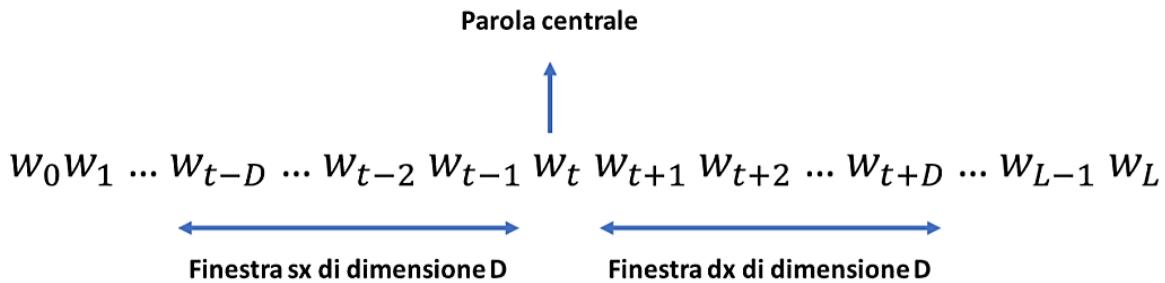


Figura 2.7: Parola centrale w_t e relativa finestra contesto

- Il modello Skip-Gram si occupa dell'addestramento di Word Embeddings attraverso l'ottimizzazione di una funzione obiettivo che calcola la probabilità delle parole contesto $\{w_{t-D}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+D}\}$ in una certa finestra di dimensione 2D data la parola centrale corrente w_t .
- Il modello Continuous Bag-of-Words si occupa dell'addestramento di Word Embeddings attraverso l'ottimizzazione di una funzione obiettivo che calcola la probabilità della parola centrale corrente data la Bag-of-Words delle parole contesto. Con il termine Bag-of Words in letteratura si viene a indicare il multi-insieme composto dalle parole w_{t-j} , dove viene ignorato l'ordine con cui tali parole compaiono nel testo ma viene mantenuta l'informazione sulla loro molteplicità.

BERT

BERT [10] (Bidirectional Encoder Representations from Transformers) è un progetto sviluppato da Google nel 2018 e consiste in un modello di rappresentazione contestuale delle parole.

BERT offre un vantaggio rispetto a modelli come Word2Vec, perché mentre in Word2Vec ogni parola ha una rappresentazione fissa indipendentemente dal contesto in cui appare, BERT produce word-embeddings che sono dinamicamente dipendenti dal contesto di parole circostanti.

BERT fornisce modelli di word embeddings pre-addestrati in moltissime lingue. Attraverso un'operazione di fine-tuning questi modelli possono poi essere utilizzati per ottenere prestazioni di altissimo livello in vari ambiti dell'NLP. I word-embeddings di BERT sono prodotti dinamicamente dal modello analizzando il contesto precedente e successivo alla parola. La rappresentazione delle parole di BERT viene sviluppata su più layer che poi vengono sommati per ottenere l'embedding finale. Per calcolare la rappresentazione finale viene infatti tenuto conto della posizione della parola all'interno della frase e del contesto di parole che circondano quella in questione.

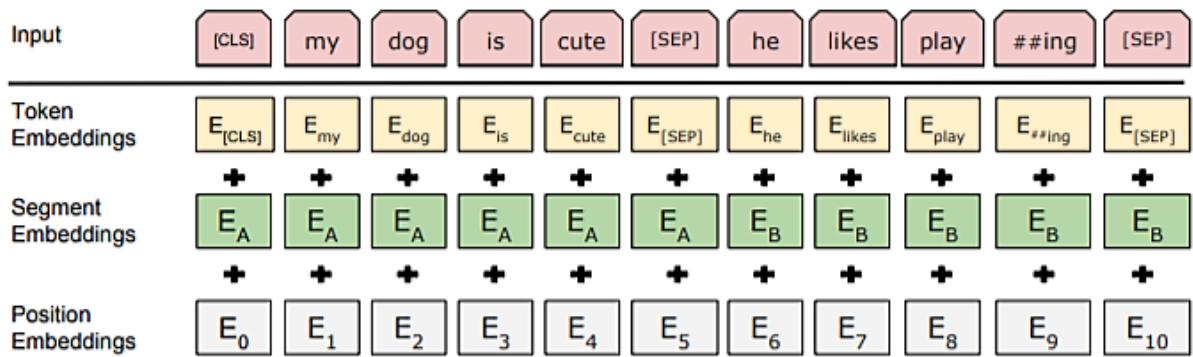


Figura 2.8: Calcolo del word-embedding secondo il language model di BERT

2.5 Algoritmi di named entity recognition

La named entity recognition (NER) è una sotto attività dell'elaborazione del linguaggio naturale con lo scopo di individuare e classificare le entità menzionate nel testo non strutturato in categorie predefinite. Le categorie predefinite possono essere ad esempio nomi di persona, organizzazioni, località, codici, espressioni temporali, quantità, valori monetari, percentuali, ecc...

Nel nostro caso andremo ad identificare e classificare informazioni personali (come nomi, età, numeri di telefono) all'interno di documenti sanitari.

Facciamo subito un esempio:

Nella seguente immagine sono state riconosciuti all'interno del testo tre differenti tipi di entità:

- “*Apple*”: è stata classificata come un'organizzazione intesa come aziende, agenzie o istituzioni.
- “*U.K.*”: è stata classificata come un'entità geopolitica intesa come città, stati o luoghi generici.
- “*\$ 1 billion*”: è stato classificato invece come un valore monetario.

Apple **ORG** is looking at buying U.K. **GPE** startup for \$1 billion **MONEY**

Figura 2.9: Esempio di Named Entity Recognition

Lo stato dell'arte per i sistemi NER per la lingua inglese producono prestazioni quasi umane. Ad esempio, uno dei migliori sistemi sviluppati per la Message Understanding Conference (MUC-7)[[12](#)], ha ottenuto il 93,39% di F-score, mentre gli annotatori umani hanno ottenuto il 97,60% e il 96,95%. Questi risultati però non dipendono solo dall'algoritmo, ma soprattutto dai dati. A seconda del problema che si sta affrontando e dello schema di annotazione scelto cambierà l'accuracy del NER. In funzione della qualità del training set e delle entità da identificare si otterranno risultati differenti.

Per poter confrontare gli algoritmi come fatto nella MUC-7 serve avere quindi condizioni paritarie.

2.5.1 Anonimizzazione in documenti di casi legali francesi

Per la valutazione delle prestazioni del nostro servizio è stato preso come riferimento un progetto [[13](#)] sviluppato da un'agenzia legale francese (Lefebvre Sarrut), in collaborazione con l'amministrazione francese (DINSIC) e la corte suprema francese (Cour de Cassation). Il progetto che hanno sviluppato si basa sull'anonimizzazione di informazioni personali all'interno di sentenze prodotte dalla corte; il tutto sempre per questioni di privacy.

Questo progetto è diventato per noi un riferimento in quanto il suo obiettivo era molto simile al nostro. Sono state infatti utilizzate le stesse tecniche basate su algoritmi di named entity recognition tramite l'uso di librerie Python per l'NLP (Spacy e Flair).

Una volta terminato lo sviluppo del nostro servizio, questo progetto è stato preso come termine di paragone per valutare le prestazioni del modello di machine learning da noi prodotto. Nel capitolo dove vengono presentati i risultati che abbiamo ottenuto è presente infatti un confronto basato sulle metriche principali. La seguente immagine rappresenta le entità e la corrispondente numerosità delle occorrenze presenti all'interno del progetto e che la corte francese ha deciso di anonimizzare.

Entity type	#
ADDRESS	5282
BAR	473
COURT	2769
DATE	12689
JUDGE_CLERK	4008
LAWYER	2915
ORGANIZATION	29987
PERS	9081
PHONE_NUMBER	178
RG	1936

Figura 2.10: Esempio di Named Entity Recognition

Capitolo 3

Definizione del problema e approccio utilizzato

La domanda che sorge spontanea e alla quale dobbiamo dare una risposta è quindi: “Qual è il metodo per poter deidentificare automaticamente i documenti sanitari?” Il nostro progetto verte sul privare i documenti sanitari dei dati identificativi, ovvero deidentificare le informazioni sensibili all’interno di un testo.

Ma come è possibile effettuare questa operazione?

Tutto questo è possibile attraverso la creazione di modelli di machine learning addestrati a partire da un dataset annotato manualmente. Per la creazione di questo servizio infatti è stato utilizzata una metodologia di approccio supervisionato.

Il punto di partenza è quindi la creazione di un dataset di referti medici annotato grazie a un particolare tool chiamato Prodigy[14]. Una volta prodotto il dataset sarà possibile fornirlo in input agli algoritmi di named entity recognition, i quali imparando dai dati addestreranno modelli in grado di rintracciare automaticamente le entità e quindi le informazioni sensibili all’interno del testo.

Ricapitolando facciamo un riassunto delle fasi principali che hanno coinvolto questo progetto, ripercorrendole per fare chiarezza:

1. Innanzitutto è stato necessario decidere quali sono le informazioni sensibili che si vogliono anonimizzare all’interno della documentazione clinica.
2. Successivamente nasce il problema della creazione di un dataset annotato da fornire in input agli algoritmi di named entity recognition. Il dataset è stato prodotto grazie al tool Prodigy, annotando dei referti medici.
3. Una volta prodotto il dataset è stato possibile addestrare un modello NER grazie alla libreria open-source Spacy.
4. Per poter migliorare il modello ottenuto con Spacy è stato necessario utilizzare i word embeddings prodotti tramite la libreria FastText.
5. Successivamente si è capito che non era più possibile migliorare nuovamente il modello ottenuto tramite l’utilizzo di Spacy e FastText. Siamo quindi passati all’addestramento di un nuovo modello NER più performante ottenuto grazie alla libreria Flair.

6. Infine è stata fatta un'accurata analisi dei risultati ottenuti, confrontando tra loro i vari modelli di named entity recognition sviluppati. E' stato inoltre confrontato il nostro best model con quanto realizzato dalla corte suprema francese all'interno del loro progetto.

3.1 Criteri di annotazione utilizzati

Il primo passo per la realizzazione di questo progetto è stato quello di decidere quali informazioni andare ad individuare all'interno di un testo sanitario per poter attuare successivamente delle strategie di anonimizzazione.

Prendendo spunto dai 18 tipi di informazioni sanitarie protette, definite all'interno della legge HIPAA, e dopo un'accurata analisi, sono state definite 16 categorie di entità da rintracciare all'interno dei documenti.

Di seguito sono presentate le entità scelte con una breve descrizione e il tag utilizzato per individuarle.

- **Personale clinico:** nomi e cognomi del personale sanitario (Dottori, Infermieri, ecc...).
 - Tag: **HCP**
- **Pazienti:** nomi e cognomi dei pazienti o dei loro parenti.
 - Tag: **NAME**
- **Età:** Età di pazienti, personale, parenti dei pazienti (ecc...).
 - Tag: **AGE**
- **Date di eventi:** ogni riferimento temporale a avvenimenti importanti.
 - Tag: **DATE**
- **Luoghi:** Luoghi specifici (Città, Luoghi di residenza, ecc...).
 - Tag: **LOC**
- **Indirizzi:** Indirizzi specifici (Via e numero civico).
 - Tag: **ADDRESS**

- **CAP:** Codice di avviamento postale
 - Tag: **ZIP**
- **Strutture:** Strutture generiche (Ospedali, Case residenziali, ecc...).
 - Tag: **FAC**
- **Nome struttura:** Nome specifico di una struttura.
 - Tag: **FAC_NAME**
- **Relazioni familiari:** relazioni familiari di pazienti (Madre, Figlio, Nonno, Nipote ecc...).
 - Tag: **REL**
- **Codice Fiscale:** codice fiscale di pazienti clinici.
 - Tag: **TAX_CODE**
- **Numero di telefono:** numeri di telefono sia fisso che cellulare.
 - Tag: **PH**
- **Orari:** Orari specifici di eventi.
 - Tag: **TIME**
- **E-mail:** Indirizzi e-mail di pazienti, personale sanitario o strutture cliniche.
 - Tag: **EMAIL**
- **URL:** Indirizzi web specifici.
 - Tag: **URL**
- **Targhe:** Targhe di autovetture sanitarie o di eventualmente di automobili appartenenti a personale/pazienti (Ambulanze, ecc...).
 - Tag: **PLATE**

Nella seguente immagine è presente un esempio della possibile annotazione di un referto medico (con informazioni inventate) annotato seguendo i criteri appena presentati.

Sig.ra **Rossi Maria** **NAME** , **anni 70** **AGE** , residente a **Modena** **LOC**
in **via Mazzini 5** **ADDRESS** . cell . **figlio** **REL** **Luigi** **NAME** **3333333333**
PH . Paziente affetta da esiti di impianto di Artroprotesi totale dell'
anca sin su coxartrosi eseguito in data **7 - 03 - 14** **DATE** presso l'
Ospedale **FAC** di **Sassuolo** **LOC** . Deambula con l' ausilio di 2
antibrachiali sec . schema a 3 tempi . Obiettivamente non segni di TVP
in atto arti inf , cicatrice chirurgica in ordine . Continua il trattamento
in regime ambulatoriale come programmato . Prossimo appuntamento
15 - 01 - 2018 **DATE** alle **11:00** **TIME** . Se ci sono novità mi scriva
alla mail **dott.verdi@gmail.com** **EMAIL** . Saluti , Dr . **Verdi** **HCP**

Figura 3.1: Esempio di annotazione secondo i criteri scelti

3.2 Creazione di un dataset

Il problema principale riscontrato all'inizio di questo progetto è che non era presente un dataset già annotato da poter utilizzare. Per poterlo produrre è stato utilizzato un tool di annotazione chiamato Prodigy. Attraverso Prodigy è stato possibile annotare 1600 referti medici, rintracciando all'interno dei testi le occorrenze delle entità presentate nella sezione precedente. Una volta terminata l'annotazione, questo tool produce un dataset in un formato compatibile con l'addestramento di un modello di named entity recognition.

3.2.1 Prodigy

Prodigy[14] è uno strumento che permette di annotare documenti o immagini in maniera semplice, veloce ed efficiente. Questo tool è stato sviluppato dal team della compagnia di software Explosion.ai, creatori fra le altre cose anche della libreria open-source di NLP chiamata SpaCy.

Prodigy consente di creare dataset annotati per la text classification, la named entity recognition e la computer vision. Attraverso una semplice interfaccia grafica, comodamente utilizzabile come applicazione web, è possibile annotare qualsiasi tipo di documento o immagine.

Prodigy permette infatti di far scorrere all'interno della sua interfaccia i documenti da annotare; in questo modo è possibile evidenziare all'interno del testo le

occorrenze delle entità, selezionando il rispettivo tag. Una volta terminata l'annotazione del documento è possibile accettarlo, rifiutarlo o ignorarlo. In questo modo si può decidere se inserire il documento appena visualizzato all'interno del dataset finale.

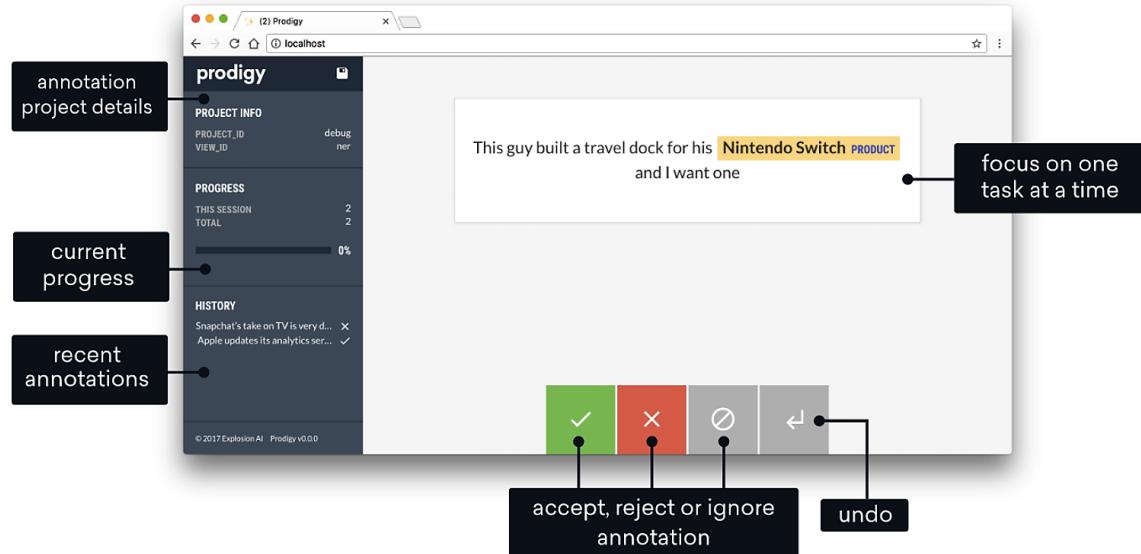


Figura 3.2: Interfaccia di Prodigy per l'annotazione dei documenti

Finita la sessione di annotazione, i documenti annotati vengono salvati nel database di Prodigy insieme alle rispettive informazioni legate alle entità rintracciate all'interno dei testi. E' quindi possibile estrarre il dataset annotato attraverso un semplice comando che permette la creazione di un file dal formato Json.

Una delle caratteristiche più importanti di Prodigy è che si integra alla perfezione con la libreria NLP SpaCy, in quanto gli sviluppatori sono i medesimi. Questo permette di poter sfruttare le numerose funzionalità di SpaCy all'interno di Prodigy, e di poter utilizzare i dataset annotati creati con Prodigy all'interno di SpaCy.

Il funzionamento di Prodigy è basato su particolari file di configurazione chiamati “*Recipes*”, scritti in Python, che permettono di definire il tipo di annotazione che si vuole svolgere e i parametri di input da utilizzare.

La modalità di annotazione che abbiamo utilizzato è stata la “*ner.manual*”, che ci ha consentito di annotare in modo manuale, ma veloce un totale di 1600 referti medici. Prodigy presenta inoltre numerose funzionalità che permettono di suddividere il dataset annotato in set di Train e set di Test, effettuare varie prove di addestramento di modelli NER e valutare i risultati ottenuti per ciascun tag attraverso le principali metriche (Precision, Recall e F-Score). Il tool consente poi di valutare la possibilità di estendere il dataset, fornendo una statistica basata sulla training-curve che mostra quanto ancora è possibile migliorare le prestazioni del modello incrementando la mole di dati in ingresso.

Le principali fasi del processo di creazione del dataset annotato sono state:

1. Pulizia del dataset iniziale composto da referti medici.
2. Annotazione dei referti medici con Prodigy attraverso l'utilizzo del seguente comando eseguito da terminale:

```
$ prodigy ner.manual dataset_referti it_core_news  
~/referti.jsonl --label NAME,HCP,AGE,DATE,LOC,  
ADDRESS,FAC, FAC_NAME,REL,TAX_CODE,PH, TIME,EMAIL,  
URL,ZIP,PLATE
```

- “*ner.manual*”: è il nome della recipe che viene utilizzata per annotare in modo manuale i referti
 - “*dataset_referti*”: è il nome del database di Prodigy in cui vengono salvate le annotazioni
 - “*it_core_news*”: è il nome del modello predefinito di SpaCy utilizzato per tokenizzare il testo da annotare all'interno dell'interfaccia di Prodigy.
 - “*~/referti.jsonl*”: è il percorso del file di referti da annotare
 - Infine sono indicati tutti i tag delle entità da annotare all'interno dei referti medici
3. Divisione del dataset annotato in set di Train e set di Test.
 4. Test di addestramento di un modello NER con Prodigy e valutazione dei primi risultati ottenuti.

La seguente tabella riporta la numerosità delle occorrenze di ciascun tag all'interno del dataset.

Questi valori risultano essere molto importanti nel momento successivo in cui andremo a effettuare una valutazione dei punteggi ottenuti dal modello in ciascuna delle entità. Ci saranno infatti entità che avendo una numerosità piuttosto bassa all'interno del dataset annotato avranno di conseguenza punteggi inferiori agli altri tag. L'algoritmo infatti avendo a disposizione pochi dati dai quali imparare risulterà essere meno performante sulle entità di minore numerosità.

La numerosità bassa di alcuni tag è dovuta al fatto che avendo annotato un dataset di referti medici, certi tipi di informazioni erano rare da trovare. Si è deciso comunque di inserire questi tag all'interno della lista di entità da anonimizzare in vista di un possibile futuro incremento della grandezza del dataset annotato con documenti di diverso tipo, i quali possono contenere queste informazioni.

Tag	Train	Test	Totale
HCP	747	356	1103
NAME	193	98	291
AGE	333	127	460
DATE	2785	1286	4071
LOC	421	208	629
FAC	225	96	351
FAC_NAME	171	72	243
REL	427	155	582
ADDRESS	56	30	86
PH	689	288	977
TIME	197	104	301
EMAIL	24	10	34
ZIP	5	1	6
URL	1	0	1
TAX_CODE	3	1	4
PLATE	1	0	1

Figura 3.3: Tabella delle numerosità di ciascuna entità all'interno del dataset

Capitolo 4

Addestramento dei modelli

Entriamo quindi nella fase cruciale del progetto, ovvero l'addestramento dei modelli di machine learning tramite l'utilizzo del dataset annotato appena creato con Prodigy.

Durante lo sviluppo di questo elaborato sono stati prodotti 3 modelli cercando di migliorare sempre di più le prestazioni e le performance.

- Il primo modello (Baseline) è stato addestrato grazie alla libreria SpaCy senza l'utilizzo di word-embeddings.
- Il modello intermedio è stato addestrato sempre tramite l'uso di SpaCy ma usufruendo di una particolare funzione che permette di inserire i word-embeddings che nel nostro caso sono stati sviluppati con la libreria FastText.
- L'ultimo modello (Best model) è stato addestrato utilizzando la libreria di NLP Flair la quale espone la possibilità di usufruire dei loro particolari word-embeddings e di poterli concatenare con quelli di FastText.

Procediamo quindi per gradi e andiamo ad analizzare i vari modelli sviluppati con tecniche e librerie Python differenti, in modo tale da poter avere una miglior panoramica del lavoro svolto.

4.1 Modello baseline sviluppato con SpaCy

Innanzitutto presentiamo che cos'è SpaCy e in seguito verrà spiegato come questa libreria è stata utilizzata per addestrare un modello NER.

4.1.1 Spacy

SpaCy[15] è una libreria open source per l'elaborazione del linguaggio naturale, scritta in Python e sviluppata dal team di Explosion.ai. La libreria è rilasciata sotto licenza MIT ed attualmente implementa modelli statistici di reti neurali in inglese, tedesco, spagnolo, portoghese, francese, italiano, olandese e greco.

SpaCy è una delle librerie più utilizzate nel campo dell'NLP in quanto offre funzionalità di tokenizzazione, part-of-speech tagging, dependency parsing, lemmatization, named entity recognition, text classification e rule-based matching.

Mentre alcune delle funzionalità di spaCy funzionano in modo indipendente, altre richiedono il caricamento di modelli statistici , che consentono a spaCy di prevedere le annotazioni linguistiche, come ad esempio se una parola è un verbo o un sostantivo.

SpaCy attualmente offre modelli statistici per una grande varietà di linguaggi, che possono essere installati come singoli moduli Python. I modelli possono differire per dimensioni, velocità, utilizzo della memoria, precisione e dati inclusi.

In genere i modelli pre-addestrati includono i seguenti componenti:

- **Pesi binari** per il POS tagging, il dependency parsing e il NER per prevedere tali annotazioni all'interno del contesto della frase.
- **File di dati** come regole di lemmatizzazione e tabelle di ricerca.
- **Vettori di parole** , ovvero rappresentazioni di multidimensionali di parole che consentono di determinare quanto siano simili tra loro.
- **Opzioni di configurazione**, come la lingua e le impostazioni della pipeline di elaborazione, per permettere a spaCy di caricare il modello correttamente.

I modelli per poter analizzare il testo utilizzano una pipeline di elaborazione composta da dei componenti che rappresentano le funzionalità enunciate in precedenza. La pipeline utilizzata dai modelli predefiniti è costituita innanzitutto da un tokenizer, poi da un tagger, da un parser e infine da un NER.

I componenti all'interno della pipeline ovviamente possono essere rimossi o sostituiti con componenti custom sviluppati dal programmatore a seconda delle necessità.

Quando viene chiamata la funzione “*nlp*” su di un documento, spaCy prima tokenizza il testo, poi lo analizza attraverso i componenti della pipeline restituendo un oggetto “*Doc*”. Tramite l'oggetto Doc è possibile quindi visualizzare le analisi effettuate sul testo dal modello durante le diverse fasi di elaborazione.

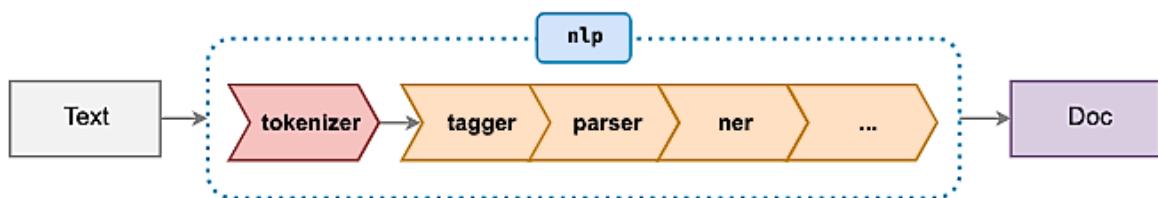


Figura 4.1: Pipeline modelli predefiniti SpaCy

4.1.2 Architettura della rete neurale

L'architettura della rete neurale di spaCy non è stata resa pubblica in maniera estesa. Durante però l'uscita della versione 2.0 della libreria uno dei due co-fondatori di

spaCy, Matthew Honnibal, ha presentato brevemente la struttura della rete neurale sulla quale si basa la named entity recognition.

Innanzitutto SpaCy dispone di una propria libreria di deep learning chiamata “*Thinc*” utilizzata per diversi modelli di NLP.

Per la maggior parte delle attività, spaCy utilizza una rete neurale profonda basata su di un'architettura caratteristica di una Convolutional Neural Network (CNN), apportando alcune modifiche.

In particolare per la NER, spaCy utilizza:

- Un approccio basato sulla transizione, preso in prestito dai parser shift-reduce, descritto nel documento Neural Architectures for Named Entity Recognition di Guillaume Lample [16].
- Un framework che Matthew Honnibal ha denominato:
"Embed. Encode. Attend. Predict" [17]

Il testo viene rappresentato da sequenze di vettori, che poi vengono codificati in una matrice di frasi da una rete RNN (Recurrent Neural Network) bidirezionale. Le righe di questa matrice possono essere viste come vettori di token sensibili al contesto della frase. Infine la matrice di frasi viene ridotta a un sentence vector che verrà utilizzato poi per la fase di predizione.

Ecco come funziona il tutto:

- **Embed:** Inizialmente il testo viene tokenizzato, poi viene creato vettore embedding della parola. All'interno del vettore vengono inserite altre informazioni riguardanti ad esempio la posizione della parola all'interno della frase.

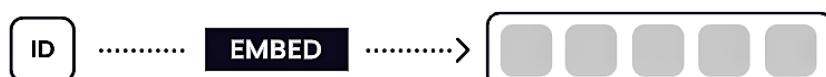


Figura 4.2: Fase di Embed

- **Encode:** Data una sequenza di vettori di parole, il passaggio di codifica calcola una rappresentazione chiamata matrice di frasi , dove ogni riga rappresenta il significato di ogni token nel contesto del resto della frase. La tecnologia utilizzata per questo scopo è un RNN bidirezionale.

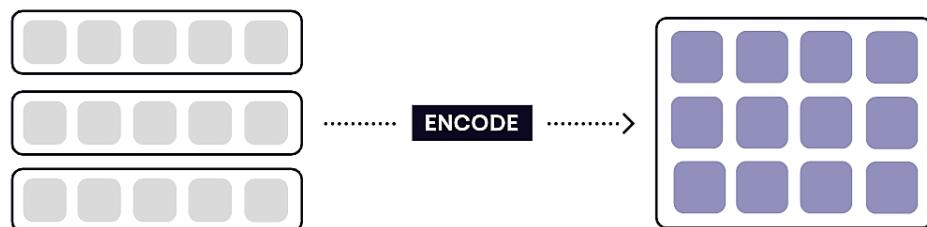


Figura 4.3: Fase di Encode

- **Attend:** La fase attend riduce la rappresentazione della matrice prodotta dalla fase di codifica a un singolo vettore, in modo che possa essere trasmessa a una rete feed-forward standard per la previsione. Il vantaggio caratteristico di un meccanismo di attend rispetto ad altre operazioni di riduzione è che prende come input anche un vettore di contesto ausiliario. Riducendo la matrice a un vettore, si stanno necessariamente perdendo informazioni; il vettore di contesto interviene in questo momento indicando quali informazioni scartare.

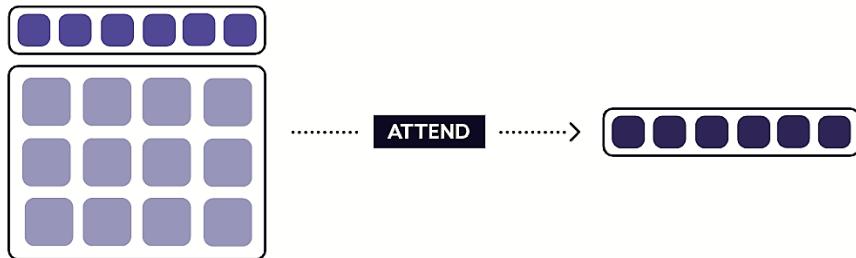


Figura 4.4: Fase di Attend

- **Predict:** Una volta che il testo è stato ridotto in un unico vettore è possibile predire la rappresentazione del target (nel nostro caso il tag corretto).

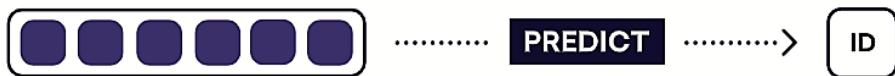


Figura 4.5: Fase di Predict

Ora però vediamo più nel dettaglio che cos’è una rete neurale convoluzionale. Nell’apprendimento automatico, una rete neurale convoluzionale [18] (CNN o dall’inglese convolutional neural network) è un tipo di rete neurale artificiale feed-forward in cui il pattern di connettività tra i neuroni è ispirato dall’organizzazione della corteccia visiva animale. I neuroni sono disposti in maniera tale da rispondere alle regioni di sovrapposizione che tassellano il campo visivo. Le reti neurali convoluzionali sono ampiamente utilizzate per sistemi di visione artificiale (computer vision), ma hanno fatto la loro comparsa anche in ambito linguistico.

Partiamo quindi con lo spiegare cosa si intende con il processo di convoluzione. Il modo più semplice per me per comprendere cos’è una convoluzione è pensarla come una funzione di finestra scorrevole applicata a una matrice.

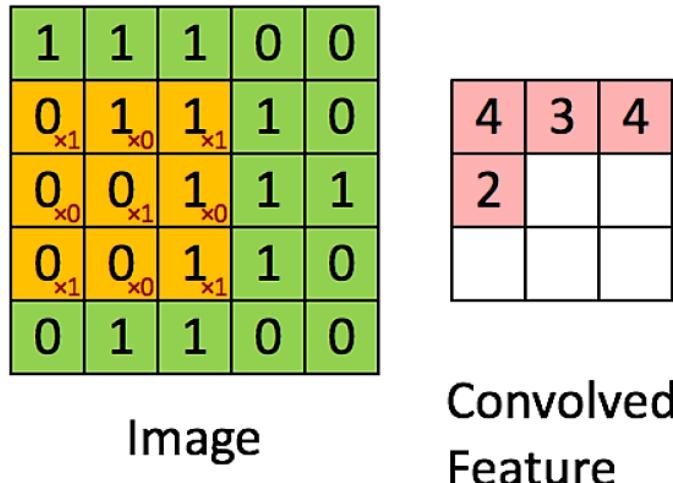


Figura 4.6: Esempio di convoluzione di un'immagine

Immaginiamo che la matrice a sinistra rappresenti un'immagine in bianco e nero. Ogni voce corrisponde a un pixel, “1” per il nero e “0” per il bianco (in genere per le immagini in scala di grigi l'intervallo è tra “0” e “255” per le immagini in scala di grigi). La finestra scorrevole è chiamata *kernel* o filtro.

Qui usiamo un filtro 3×3 , moltiplichiamo i suoi valori in termini di elemento con la matrice originale, quindi li sommiamo. Per ottenere la convoluzione completa, lo facciamo per ogni elemento facendo scorrere il filtro sull'intera matrice.

Tutto questo è utile per esempio per rilevare i bordi all'interno dell'immagine prendendo la differenza tra un pixel e i suoi vicini.

Le CNN sono fondamentalmente diversi strati di convoluzioni con funzioni di attivazione non lineari come la *ReLU* o *tanh* applicate ai risultati.

In una rete neurale feedforward tradizionale collegiamo ogni neurone di input a ciascun neurone di output nello strato successivo. Nelle CNN non lo facciamo.

Usiamo invece le convoluzioni sul livello di input per calcolare l'output. Ciò si traduce in connessioni locali, in cui ogni regione dell'input è collegata a un neurone nell'output. Ogni livello applica filtri diversi, in genere centinaia o migliaia, e combina i loro risultati.

Durante la fase di training, una CNN apprende automaticamente i valori dei propri filtri in base all'attività che si desidera eseguire. Ad esempio, nella classificazione delle immagini una CNN può imparare a rilevare i bordi dai pixel grezzi nel primo livello, quindi utilizzare i bordi per rilevare le forme semplici nel secondo livello e quindi utilizzare queste forme per individuare le forme del viso negli strati superiori. L'ultimo livello è quindi un classificatore che utilizza queste funzionalità di alto livello per individuare i volti all'interno delle immagini.

Ma quindi come si applica tutto questo al NLP?

Invece dei pixel dell'immagine, l'input per la maggior parte delle attività di NLP sono frasi o documenti rappresentati come una matrice. Ogni riga della matrice

corrisponde a un token, tipicamente una parola o un carattere, rappresentato sotto forma di un vettore (word-embeddings).

Per una frase di 10 parole che utilizza dei word-embeddings a 100 dimensioni avremo una matrice 10×100 come input. Questa è la nostra "immagine".

Una CNN per il testo opera esclusivamente su due dimensioni, dato che i filtri devono muoversi solo rispetto all'asse temporale.

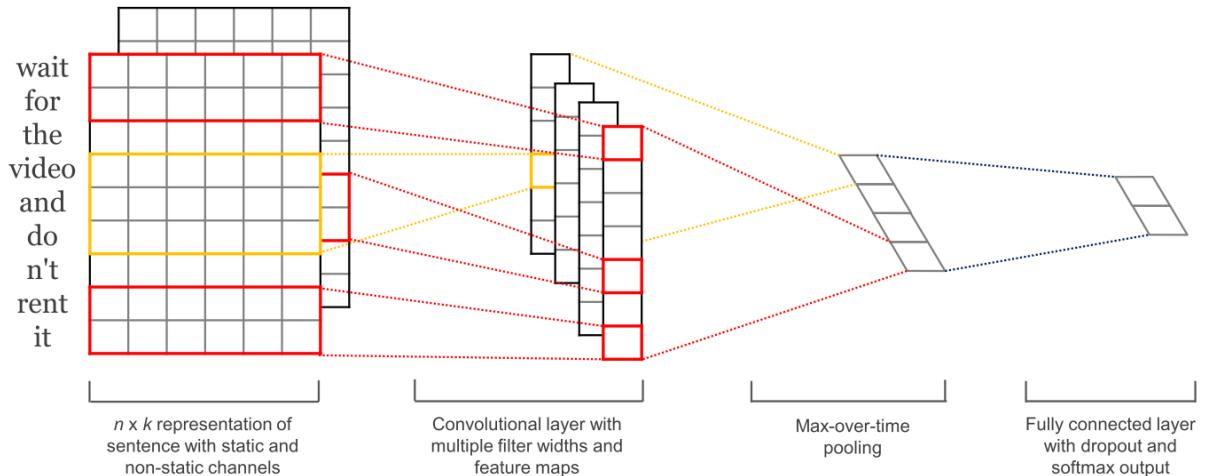


Figura 4.7: Struttura di una CNN per il NLP

Un aspetto chiave delle reti neurali convoluzionali sono i livelli di pooling, tipicamente applicati dopo i livelli convoluzionali. I layer in pool sottocampionano il loro input. Una proprietà del pooling è che fornisce una matrice di output di dimensioni fisse, che in genere è richiesta per la classificazione. Ciò ti consente di utilizzare frasi di dimensioni variabili e filtri di dimensioni variabili, ma ottieni sempre le stesse dimensioni di output da inserire in un classificatore.

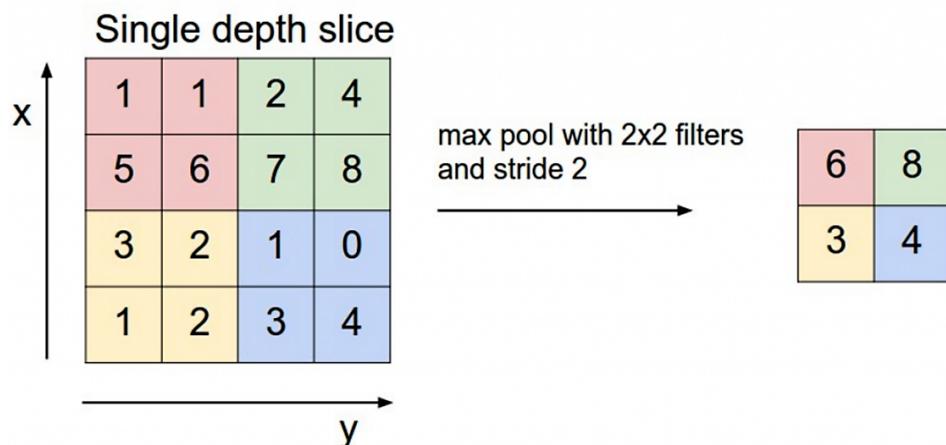


Figura 4.8: Esempio di pooling (sottocampionamento) di una matrice

Nel caso della named entity recognition il processo di classificazione consiste nel applicare un tag a ciascun token presente all'interno della frase.

SpaCy utilizza uno schema di classificazione chiamato BILOU. Oltre ad applicare uno dei tag di classificazione (Name, Date, Phone, ecc...), SpaCy attribuisce anche un ruolo del token all'interno dell'entità. Mi spiego meglio attraverso un esempio. Se un'entità rintracciata all'interno del testo corrisponde al nome “*Tizio Caio*”, il tokenizzatore dividerà questa entità in due token differenti. L'entità “*Name*” corrispondente però dovrà essere rappresentata come unica. SpaCy quindi attribuisce una sigla oltre al tag per stabilire la posizione del token all'interno dell'entità in questione.

Lo schema che adotta spaCy è il seguente:

- **B**egin: rappresenta il primo token di un'entità multi-token
- **I**n: rappresenta un token interno di un'entità multi-token
- **L**ast: rappresenta l'ultimo token di un'entità multi-token
- **O**ut: rappresenta un token che non fa parte di nessuna entità
- **U**nit: rappresenta un'entità formata da un singolo token

Nel caso dell'esempio fatto in precedenza la classificazione sarebbe quindi:

- “*Tizio*”: B - NAME
- “*Caio*”: L - NAME

4.1.3 Addestramento del modello e risultati ottenuti

L'addestramento del modello NER è stato sviluppato direttamente all'interno di Prodigy utilizzando le funzionalità di training di SpaCy.

Il training è stato effettuato tramite la comoda interfaccia Command Line di Prodigy, partendo dal modello predefinito “*Blank: it*” di SpaCy, e utilizzando il seguente comando:

```
$ prodigy train ner train_dataset blank:it --output  
~/model/directory --eval-id test_dataset
```

Il comando precedente addestra un modello NER prendendo in input il set di training “*train_dataset*” e lo valuta attraverso il set test “*test_dataset*”.

Il modello prodotto è stato allenato per un totale di 10 epochs.

Nella tabella seguente sono presenti i risultati, basati sulle metriche principali, ottenuti dal modello durante la fase di test.

Precision	Recall	F-Score
0,88	0,88	0,88

Nella seguente tabella invece sono inseriti i risultati ottenuti dal modello su ciascun tipo di entità.

Tag	Precision	Recall	F1-Score
HCP	0.95	0.96	0.95
NAME	0.92	0.75	0.83
AGE	0.90	0.73	0.81
DATE	0.85	0.90	0.87
LOC	0.92	0.85	0.88
FAC	0.81	0.80	0.80
FAC_NAME	0.88	0.75	0.81
REL	0.88	0.88	0.88
ADDRESS	0.90	0.90	0.90
PH	0.98	0.99	0.98
TIME	0.95	0.91	0.93
EMAIL	1.00	0.90	0.95
ZIP	1.00	1.00	1.00
URL	0.00	0.00	0.00
TAX_CODE	0.00	0.00	0.00
PLATE	0.00	0.00	0.00

Figura 4.9: Risultati modello Baseline

I risultati ottenuti dal modello baseline sono ritenuti abbastanza buoni, tuttavia si è constatato che attraverso tecniche differenti si può arrivare ad un risultato migliore.

4.2 Modello intermedio sviluppato con SpaCy e FastText

Il modello baseline che abbiamo ottenuto è un buon punto di partenza; attraverso l'utilizzo di una particolare funzionalità implementata da SpaCy è stato però possibile incrementare le prestazioni del modello di alcuni punti percentuale.

La funzionalità citata in precedenza si chiama SpaCy Pretrain e permette di inizializzare i layer della rete neurale convoluzionale (CNN) di SpaCy con un layer custom formato da vettori contestuali. Questo consente di inserire un livello di contesto in più all'interno della rete che porta a un miglioramento dell'accuratezza

del modello. Il comando SpaCy Pretrain permette di sfruttare il transfer learning per inizializzare il modello attraverso l'uso del testo grezzo, utilizzando un language model simile a quello del sistema BERT di Google.

Nello specifico viene caricato un modello di word-embeddings, che nel nostro caso è stato sviluppato tramite la libreria FastText, e viene addestrato un componente CNN in modo tale che possa prevedere i vettori che corrispondono a quelli pre-addestrati. Una volta terminato l'addestramento di questo componente, i pesi corrispondenti vengono salvati e possono essere utilizzati durante la fase di training di un modello di named entity recognition.

4.2.1 FastText

FastText è una libreria open-source per la rappresentazione e la classificazione del testo sviluppata dal Facebook's AI Research (FAIR) Lab.[\[9\]](#)

FastText consente la creazione di modelli di word-embeddings partendo da un corpus di testo grezzo.

Una delle caratteristiche principali di questa libreria è quella di essere leggera ed efficiente, in pochi minuti permette infatti di addestrare un modello di vettori di parole da zero. Inoltre fornisce modelli pre-addestrati di word-embeddings per 157 tipi di linguaggi differenti.

I modelli vettoriali di FastText sono basati su un'estensione dell'architettura di Word2Vec con qualche differenza. La differenza principale con Word2Vec è che FastText permette di rappresentare le parole come composizione di n-grammi di caratteri. Quindi il vettore per una parola è costituito dalla somma di n-grammi.

Ad esempio, il vettore della parola "*apple*" è la somma dei vettori dei seguenti n-grammi: <"*ap*", "*app*", "*appl*", "*apple*">.

Questo permette di generare word embeddings migliori per le parole più rare; il vettore che rappresenta queste parole può infatti essere formato da n-grammi di parole presenti più spesso all'interno del corpus del testo.

Inoltre se una parola non è presente nel vocabolario formato durante la fase di training, il suo word embedding può comunque essere formato da n-grammi di altre parole. Fasttext risulta quindi essere migliore soprattutto nei casi in cui si ha un dataset di training ridotto o formato da molte parole rare.

Questa libreria fornisce la possibilità di creare modelli di word-embeddings sia con un'architettura skip-gram che con un'architettura continuous bag-of-words. Per le finalità del nostro progetto si è però visto che il modello skip-gram risulta essere più efficace e più performante.

4.2.2 Addestramento del modello e risultati ottenuti

Anche in questo caso l'addestramento del modello NER è stato sviluppato direttamente all'interno di Prodigy utilizzando le funzionalità di training di SpaCy.

A differenza di prima però è stato necessario innanzitutto addestrare un modello di word-embeddings con FastText.

FastText permette di addestrare word-embeddings di dimensioni differenti, nel nostro caso, dopo vari test effettuati, la scelta è ricaduta sul valore 200.

Attraverso il seguente script Python è stato possibile quindi addestrare un modello skip-gram di word-embeddings di dimensione 200 partendo dal corpus di testo grezzo formato dai referti medici.

```
import fasttext  
  
model = fasttext.train_unsupervised('~/referti.txt',"skipgram", dim=200)
```

Una volta effettuato il training del modello di word-embeddings di FastText è stato possibile utilizzarlo fornendolo in input alla funzione Pretrain di SpaCy.

Attraverso il seguente comando è stato addestrato il componente di pretrain:

```
$ spacy -m pretrain ~/referti.jsonl fasttext_model.bin ~/output_directory
```

Tramite il comando precedente vengono presi in input il testo grezzo in formato Jsonl (Json Lines) e il modello di FastText e viene addestrato il layer di pretrain. Successivamente è stato svolto il training del modello NER partendo dal dataset annotato e inserendo il componente di pretrain.

```
$ prodigy train ner train_dataset blank:it --output  
~/model/directory --eval-id test_dataset --init-tok2vec  
~/pretrain_model
```

I parametri di training sono i medesimi della baseline, l'unica differenza sta nell'inizializzazione del modello tramite i pesi prodotti grazie alla funzionalità Pretrain di SpaCy. Questi pesi vengono inseriti nel comando di train tramite il parametro --init-tok2vec.

Nella tabella seguente sono presenti i risultati, ottenuti dal modello intermedio durante la fase di test.

Precision	Recall	F-Score
0,89	0,89	0,89

Nella seguente tabella invece sono inseriti i risultati ottenuti dal modello su ciascun tipo di entità.

Tag	Precision	Recall	F1-Score
HCP	0.94	0.94	0.94
NAME	0.90	0.71	0.79
AGE	0.88	0.75	0.80
DATE	0.85	0.89	0.87
LOC	0.90	0.83	0.86
FAC	0.84	0.79	0.81
FAC_NAME	0.92	0.78	0.84
REL	0.85	0.87	0.86
ADDRESS	0.87	0.87	0.87
PH	0.98	0.98	0.98
TIME	0.92	0.90	0.91
EMAIL	1.00	0.70	0.82
ZIP	1.00	1.00	1.00
URL	0.00	0.00	0.00
TAX_CODE	0.00	0.00	0.00
PLATE	0.00	0.00	0.00

Figura 4.10: Risultati modello Intermedio

Come si può vedere dalle tabelle, attraverso l'utilizzo della feature Pretrain e quindi dei word-embeddings, il modello risulta ottenere prestazioni migliori rispetto alla baseline.

4.3 Modello finale sviluppato con Flair e FastText

Il modello intermedio ottenuto con SpaCy e FastText ha portato a un buon miglioramento in termini di prestazioni.

Analizzando la letteratura e lo stato dell'arte degli algoritmi di named entity recognition è stato possibile visualizzare i risultati che si possono ottenere con una libreria simile a SpaCy, chiamata Flair. Attraverso l'utilizzo di Flair è stato prodotto un ultimo modello NER, che confrontato con quelli sviluppati con SpaCy ha conseguito risultati e prestazioni migliori.

Presentiamo quindi questa libreria e le fasi che hanno portato all'addestramento del nostro modello gold-standard.

4.3.1 Flair

Flair è un framework state-of-art per l'elaborazione del linguaggio naturale sviluppato all'Università Humboldt di Berlino in collaborazione con il team di Zalando Research.

Questa libreria Python dispone di numerose funzionalità per il NLP come la named entity recognition, part-of-speech tagging, text classification e sense disambiguation. Inoltre Flair, basando l'architettura dei suoi algoritmi sull'utilizzo dei word-embeddings, espone funzionalità per l'addestramento e la combinazione di vari tipi di modelli vettoriali per la rappresentazione delle parole.

Attraverso l'utilizzo di Flair è infatti possibile fare il fine-tuning di modelli di word-embeddings pre-addestrati, utilizzando un corpus di testo grezzo. Inoltre una delle altre feature molto interessanti è la possibilità di combinare i word-embeddings prodotti tramite varie librerie impilandoli all'interno di uno stack. Per esempio si possono concatenare i vettori sviluppati tramite BERT, Word2Vec o FastText con quelli prodotti utilizzando Flair.

I Contextual string embeddings di Flair catturano informazioni sintattico-semantiche che vanno oltre i word-embeddings standard.

Le differenze chiave sono che:

- sono addestrati senza alcuna nozione esplicita di parola e quindi fondamentalmente modellano le parole come sequenze di caratteri.
- sono contestualizzati dal testo circostante, il che significa che la stessa parola avrà embeddings diversi a seconda del suo utilizzo contestuale.

I modelli di word-embeddings pre-addestrati da Flair si possono distinguere in due particolari tipologie: forward e backward. I modelli forward cercano di predire le parole successive a quella in questione all'interno della frase. I modelli backward invece cercano di predire le parole antistanti la parola in questione.

Utilizzando entrambi questi modelli attraverso una concatenazione dei vettori che rappresentano le parole sarà possibile fornire all'algoritmo di NER un contesto più ampio, e di conseguenza si potranno ottenere risultati migliori.

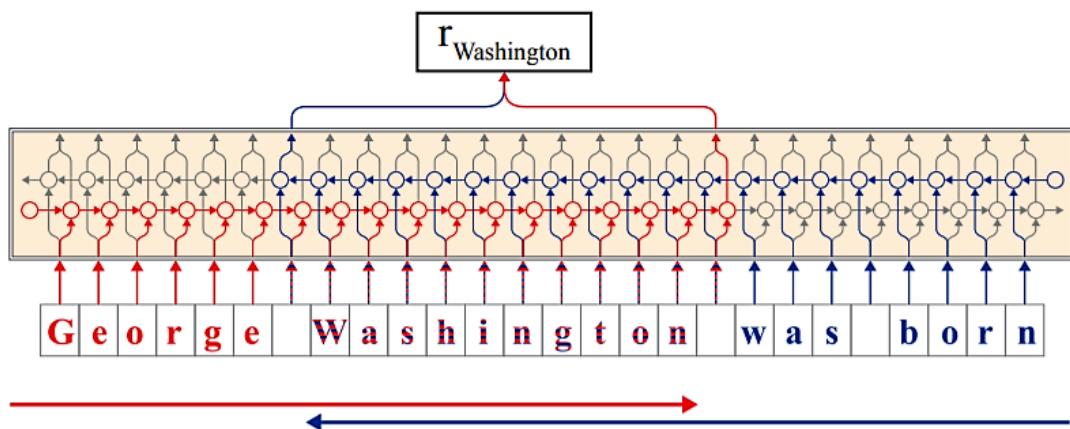


Figura 4.11: Modelli forward e backward per la predizione dei word-embeddings

4.3.2 Architettura della rete neurale

L'architettura della rete neurale di Flair è stata descritta all'interno del documento “*Contextual String Embeddings for Sequence Labeling*” by Alan Akbik, Duncan Blythe, Roland Vollgraf [11].

Il tipo di rete neurale utilizzata per gli algoritmi di NLP può essere rappresentata e descritta attraverso la seguente figura.

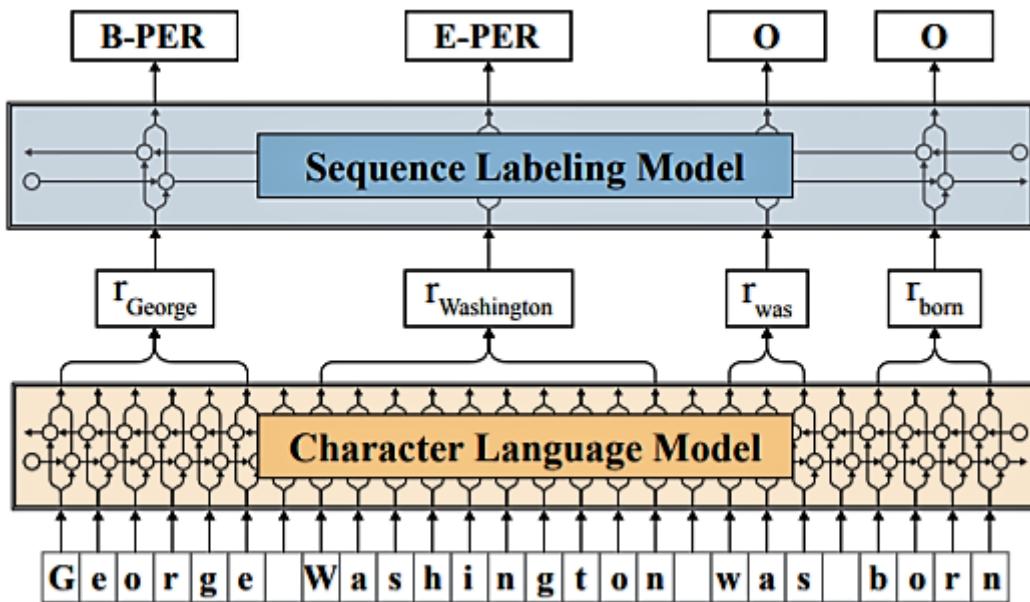


Figura 4.12: Architettura della rete neurale di Flair per la NER

Per la rappresentazione delle parole Flair utilizza un *Character Language Model* basato sulla variante LSTM di una rete neurale ricorrente (RNN).

Analizziamo quindi questo tipo di rete neurale.

Una RNN mantiene un memoria basata su informazioni cronologiche, che consente al modello di prevedere l'output corrente condizionato da informazioni passate.

A volte, la capacità di queste ultime di avere una memoria è molto ridotta.

Le reti LSTM, introdotte da Hochreiter e Schmidhuber nel 1997 [19], risolvono esattamente questo problema.

LSTM sta per *Long short-term memory* ed è una rete artificiale neurale ricorrente utilizzata in particolare nel campo del deep-learning. A differenza delle reti neurali feedforward standard , LSTM ha connessioni di feedback.

Un'unità LSTM comune è composta da una cella , un gate di ingresso , un gate di uscita e un gate di forget . La cella ricorda i valori su intervalli di tempo arbitrari e le tre porte regolano il flusso di informazioni in entrata e in uscita dalla cella.

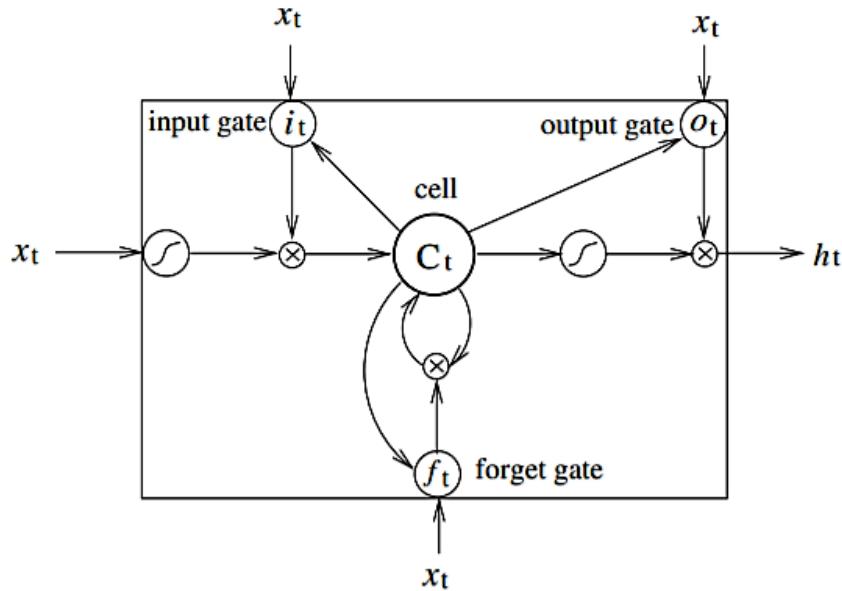


Figura 4.13: Struttura di un cella LSTM

Il vantaggio di una cella LSTM rispetto a una comune unità ricorrente è la sua unità di memoria cellulare. Il vettore cellulare ha la capacità di incapsulare l'idea di dimenticare una parte della sua memoria precedentemente immagazzinata, così come di aggiungere delle nuove informazioni.

Le reti LSTM sono adatte per classificare , elaborare e fare previsioni basate su dati di serie temporali , poiché possono esserci ritardi di durata sconosciuta tra eventi importanti.

Attraverso il *Character Language Model* vengono prodotti i Contextual string embeddings, di cui parlavamo nella sezione precedente. Questi poi vengono passati allo strato superiore della rete neurale, il *Sequence labeling model*.

Il *Sequence labeling model* è basato su una struttura descritta all'interno del paper “Bidirectional LSTM-CRF Models for Sequence Tagging” by Huang et al. (2015)[[20](#)]. Questo tipo di architettura è basata su un modello bidirezionale LSTM con un conditional random field (CRF) decoding layer.

Nell'attività di sequence tagging abbiamo accesso a feature di input passate e future per un determinato periodo di tempo, possiamo quindi utilizzare una rete LSTM bidirezionale (BI-LSTM). Così facendo, possiamo utilizzare in modo efficiente le feature passate (tramite gli stati forward) e le feature future (tramite stati a ritroso) per un periodo di tempo specifico, il tutto per mezzo la backpropagation.

Per poter utilizzare le informazioni sui tag adiacenti nella previsione dei tag correnti è poi stato inserito un CRF layer. È stato dimostrato infatti che i CRF possono in generale produrre tagging con un'accuratezza più alta.

Attraverso la combinazione di una rete BI-LSTM con un layer CRF si ottiene quindi un'architettura chiamata BI-LSTM-CRF.

Nella seguente immagine possiamo visualizzare questo tipo di architettura. Le celle LSTM sono rappresentate attraverso i quadratini barrati con i bordi stoncati.

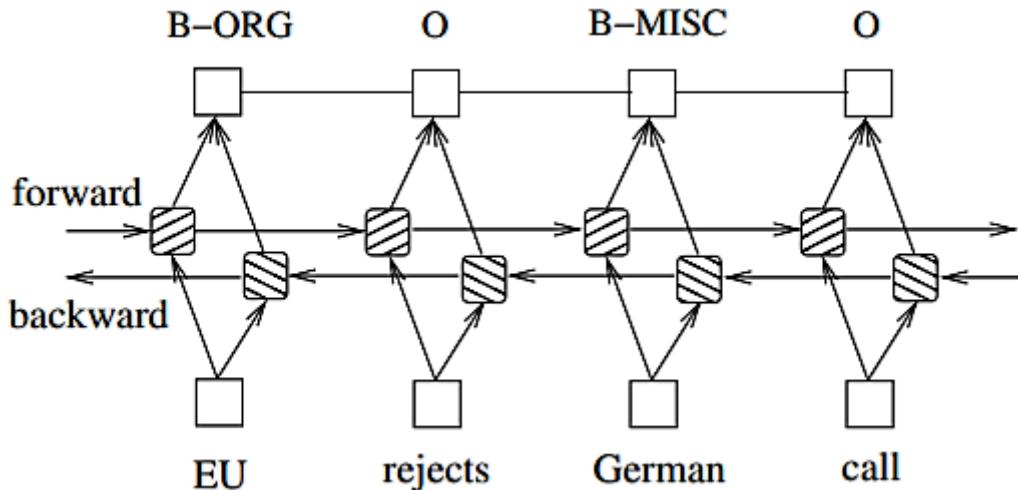


Figura 4.14: Architettura di una rete BI-LSTM-CRF

A differenza di SpaCy, Flair utilizza uno schema di classificazione più generale e semplice chiamato BIO. Vengono infatti rimossi i tag Last e Unit.

Lo schema adottato da Flair è il seguente:

- **B**egin: rappresenta il primo token di un’entità multi-token
- **I**n: rappresenta un token interno di un’entità multi-token
- **O**ut: rappresenta un token che non fa parte di nessuna entità

4.3.3 Addestramento del modello e risultati ottenuti

Innanzitutto è stato fatto il fine-tuning dei modelli di word-embeddings pre-addestrati (*it-backward* e *it-forward*) utilizzando il testo grezzo di referti medici che avevamo a disposizione.

```
from flair.data import Dictionary
from flair.embeddings import FlairEmbeddings
from flair.trainers.language_model_trainer import LanguageModelTrainer, TextCorpus

model_name = 'it-forward'

# instantiate an existing LM, such as one from the FlairEmbeddings
language_model = FlairEmbeddings(model_name).lm
is_forward_lm = language_model.is_forward_lm
```

```

# get the dictionary from the existing language model
dictionary: Dictionary = language_model.dictionary

# get your corpus, process forward and at the character level
corpus = TextCorpus('corpus',
                     dictionary,
                     is_forward_lm,
                     character_level=True)

# use the model trainer to fine-tune this model on your corpus
trainer = LanguageModelTrainer(language_model, corpus)

trainer.train(f'flair/lm/{model_name}',
              sequence_length=100,
              mini_batch_size=100,
              learning_rate=20,
              patience=10,
              checkpoint=True)

```

Una volta ottenuti i modelli di word-embeddings forward e backward è stato possibile addestrare il NER.

Per poter ottenere vettori contestuali sempre migliori il modello NER è stato addestrato tramite l'utilizzo della funzionalità Stacked-embeddings di Flair. Questa libreria permette infatti di impilare i word-embeddings all'interno di uno stack. Effettuando questa operazione le parole vengono rappresentate da un unico vettore PyTorch formato dalla concatenazione degli embedding di vario tipo. Nel nostro caso abbiamo inserito all'interno dello stack i modelli forward e backward di Flair e il modello sviluppato con la libreria FastText. Il risultato finale per la rappresentazione di una parola è dunque un unico vettore Pytorch di dimensione 4296, formato dalla concatenazione degli embedding di Flair (dimensione 2048 ciascuno) con quelli di FastText (dimensione 200).

In letteratura sono state cercate anche altre tecniche per la combinazione di word-embeddings di tipo differente. Una delle altre tecniche più utilizzate è ad esempio fare una media pesata dei word-embeddings. Tuttavia però, come si può dedurre dai seguenti paper [23][24][25], i risultati che si possono ottenere facendo una media pesata dei vettori sono molto simili a quelli ottenibili tramite il processo di concatenazione.

L'addestramento del modello NER è stato effettuato quindi attraverso l'utilizzo del seguente script Python.

```
from flair.data import Corpus
```

```

from flair.datasets import ColumnCorpus
from flair.embeddings import FlairEmbeddings, StackedEmbeddings,
TokenEmbeddings, FastTextEmbeddings
from typing import List
from flair.trainers import ModelTrainer
from flair.models import SequenceTagger

# define columns
columns = {0 : 'text', 1 : 'ner'}
# directory where the data resides
data_folder = '~/deidentificazione'
# initializing the corpus
corpus: Corpus = ColumnCorpus(data_folder, columns,
                                train_file = 'train.txt',
                                test_file = 'test.txt',
                                dev_file = 'dev.txt')

# tag to predict
tag_type = 'ner'
# make tag dictionary from the corpus
tag_dictionary = corpus.make_tag_dictionary(tag_type=tag_type)

# Define the different word-embeddings
forward_embs= FlairEmbeddings('~/deidentificazione/it-forward/best.pt')
backward_embs=FlairEmbeddings('~/deidentificazione/it-backward/best.pt')
fasttext_embs= FastTextEmbeddings('~/deidentificazione/fasttext_model.bin')

# Insert the different word-embeddings in the Stack
embedding_types : List[TokenEmbeddings] = [forward_embs, backward_embs,
fasttext_embs]

embeddings : StackedEmbeddings = StackedEmbeddings(
                                embeddings=embedding_types)

# Define the SequenceTagger
tagger : SequenceTagger = SequenceTagger(hidden_size=256,
                                           embeddings=embeddings,
                                           tag_dictionary=tag_dictionary,
                                           tag_type=tag_type,
                                           use_crf=True)

# Define the trainer and start the training of the model
trainer : ModelTrainer = ModelTrainer(tagger, corpus)
trainer.train('~/deidentificazione/ner_model',
             learning_rate=0.1,
             mini_batch_size=32,
             max_epochs=30)

```

Nella tabella seguente sono presenti i risultati, ottenuti durante la fase di test dal modello sviluppato con Flair.

Precision	Recall	F-Score
0.91	0.93	0.92

Nella seguente tabella invece sono inseriti i risultati ottenuti dal modello su ciascun tipo di entità.

Tag	Precision	Recall	F1-Score
HCP	0.95	0.97	0.96
NAME	0.91	0.87	0.89
AGE	0.90	0.95	0.92
DATE	0.89	0.92	0.90
LOC	0.94	0.92	0.93
FAC	0.87	0.82	0.84
FAC_NAME	0.87	0.87	0.87
REL	0.84	0.91	0.87
ADDRESS	0.90	0.90	0.90
PH	0.99	0.99	0.99
TIME	0.94	0.99	0.96
EMAIL	1.00	1.00	1.00
ZIP	1.00	1.00	1.00
URL	0.00	0.00	0.00
TAX_CODE	0.00	0.00	0.00
PLATE	0.00	0.00	0.00

Figura 4.15: Risultati modello finale

Come si può vedere dalle tabelle, attraverso l'utilizzo del framework Flair e degli stacked-embeddings, il modello risulta ottenere prestazioni migliori rispetto ai due precedenti.

Capitolo 5

Analisi dei risultati e confronto tra i modelli

Ricapitolando, abbiamo creato un dataset annotato ed effettuato l'addestramento di 3 differenti modelli di machine learning per la Named Entity Recognition. Ora non ci rimane altro che effettuare un'analisi dei risultati ottenuti e un confronto tra i modelli sviluppati.

5.1 Confronto tra i modelli

Attraverso la seguente tabella è possibile confrontare i risultati ottenuti dai vari modelli NER ottenuti con tecniche differenti.

Modello	Precision	Recall	F-Score
Spacy Baseline	0,88	0,88	0,88
Spacy + FastText	0,89	0,89	0,89
Flair + FastText	0,91	0,93	0,92

Come si può vedere, il modello sviluppato con Flair è il migliore dei tre in tutte le metriche analizzate. Tra il modello baseline e il modello intermedio invece, non risulta esserci una grandissima differenza in termini di risultati. Si nota qualche piccolo miglioramento infatti, solamente analizzando le cifre decimali dopo la virgola.

Tra le metriche principali che abbiamo utilizzato per valutare i modelli, la Recall è quella che ci interessa di più. Questo perchè, essendo l'anonymizzazione lo scopo principale del progetto, risulta essere più importante individuare un maggior numero di entità magari sbagliandone qualcuna, rispetto magari a lasciarne scappare alcune avendo una precisione più alta.

5.2 Match e Overlap

Per poter effettuare un'analisi più accurata dei risultati ottenuti dal nostro modello gold-standard sono state calcolate le metriche principali basandosi non solo su un match esatto dell'entità, ma anche nel caso in cui non ci fosse una corrispondenza completa tra l'entità individuata dal modello e quella corretta definita dall'annotazione.

Mi spiego meglio, in alcuni casi il modello individua solo una parte dell'entità che dovrebbe individuare, questo nel processo di valutazione corrisponde ad un errore. Stessa cosa vale nel momento in cui vengono individuate insieme all'entità corretta parti di testo che non c'entrano. Nel momento in cui si va a fare l'analisi dei risultati questo tipo di errori pesano. Tuttavia, in certi casi, per lo scopo di anonimizzazione questi tipi di errori potrebbero anche essere contemplati.

Sono quindi state ricalcate le metriche principali tenendo conto anche delle individuazioni parziali delle entità da parte del modello.

Attraverso le seguenti tabelle si possono confrontare i risultati a seconda di un'individuazione precisa (Match) o parziale (Overlap) dell'entità all'interno del testo.

Match

Tag	Precision	Recall	F1-Score
HCP	0.95	0.97	0.96
NAME	0.91	0.87	0.89
AGE	0.90	0.95	0.92
DATE	0.89	0.92	0.90
LOC	0.94	0.92	0.93
FAC	0.87	0.82	0.84
FAC_NAME	0.87	0.87	0.87
REL	0.84	0.91	0.87
ADDRESS	0.90	0.90	0.90
PH	0.99	0.99	0.99
TIME	0.94	0.99	0.96
EMAIL	1.00	1.00	1.00
ZIP	1.00	1.00	1.00

Figura 5.1: Risultati ottenuti dal modello secondo un Match completo

Overlap

Tag	Precision	Recall	F1-Score
HCP	1,00	1,00	1,00
NAME	1,00	0,96	0,98
AGE	0,98	0,98	0,98
DATE	0,92	0,96	0,94
LOC	0,97	0,96	0,97
FAC	0,90	0,84	0,87
FAC_NAME	0,92	0,97	0,94
REL	0,86	0,92	0,89
ADDRESS	1,00	1,00	1,00
PH	1,00	1,00	1,00
TIME	0,98	0,98	0,98
EMAIL	1,00	1,00	1,00
ZIP	1,00	1,00	1,00

Figura 5.2: Risultati ottenuti dal modello secondo l'Overlap

Ovviamente tenendo conto di questo particolare aspetto i risultati ottenuti dal modello con l'Overlap sono nettamente migliori rispetto a quelli che si ottengono secondo l'utilizzo del Match completo.

5.3 Confronto con il progetto della corte suprema francese

Il progetto di anonimizzazione di casi legali francesi [21][22] è stato preso come riferimento anche per la valutazione dei risultati che abbiamo ottenuto.

Ovviamente le entità da anonimizzare non sono le stesse, tuttavia il modello francese può essere un buon termine di paragone per fare una valutazione delle prestazioni del nostro modello NER.

Nelle seguenti figure sono messi a confronto i risultati ottenuti per ciascuna entità dal nostro modello migliore (Best model) con quelli ottenuti dal modello del team di sviluppo francese.

Entity	Precision	Recall	F1
ADDRESS	0.8876	0.8639	0.8756
BAR	1.0000	1.0000	1.0000
COURT	0.9216	0.9495	0.9353
DATE	0.9823	0.9808	0.9815
JUDGE_CLERK	0.9124	0.9605	0.9358
LAWYER	0.9495	0.9617	0.9556
ORGANIZATION	0.9503	0.9324	0.9413
PERS	0.9570	0.9408	0.9488
PHONE_NUMBER	0.9583	0.9200	0.9388
RG	0.9122	0.9353	0.9236

Figura 5.3: Risultati ottenuti dal modello francese per ciascuna entità

Tag	Precision	Recall	F1-Score
HCP	0.95	0.97	0.96
NAME	0.91	0.87	0.89
AGE	0.90	0.95	0.92
DATE	0.89	0.92	0.90
LOC	0.94	0.92	0.93
FAC	0.87	0.82	0.84
FAC_NAME	0.87	0.87	0.87
REL	0.84	0.91	0.87
ADDRESS	0.90	0.90	0.90
PH	0.99	0.99	0.99
TIME	0.94	0.99	0.96
EMAIL	1.00	1.00	1.00
ZIP	1.00	1.00	1.00

Figura 5.4: Risultati ottenuti dal nostro modello per ciascuna entità

Come si può vedere dalle tabelle, i risultati ottenuti dal nostro modello si allineano all'incirca a quelli ottenuti dal modello francese. Come già detto, non è possibile effettuare un confronto paritario tra tutte le entità; è però possibile paragonare i tag che sono presenti in entrambi i modelli.

Per quanto riguarda ad esempio il tag PERS del modello francese, che nel nostro caso corrisponde all'entità NAME, il loro modello risulta avere risultati migliori.

Uno dei motivi principali potrebbe essere il fatto che nel nostro caso la numerosità di questo tipo di tag è molto inferiore (200 occorrenze) rispetto alla loro (9081 occorrenze).

Per quanto concerne invece i tag ADDRESS e PHONE_NUMBER il nostro modello risulta avere risultati migliori rispetto al modello sviluppato dalla corte suprema francese.

Capitolo 6

Visualizzazione predizioni del modello

Uno dei fattori che mancavano a questo progetto era il poter visualizzare in modo comodo ed efficace le predizioni del modello direttamente sui documenti. Tutto questo è stato reso possibile attraverso l'utilizzo di Prodigy.

Ma come è stato possibile integrare un modello sviluppato con la libreria Flair all'interno dell'ecosistema SpaCy?

6.1 Custom Recipe

Innanzitutto è stato sviluppato appositamente un componente NER custom per poter inserire il modello Flair all'interno della pipeline SpaCy.

Il componente NER custom è il seguente:

```
class Custom_Ner(object):
    name = "flair_ner"

    def __init__(self, nlp, model):
        self.tagger = model

    def __call__(self, doc):
        sentence = Sentence(doc.text, use_tokenizer=SpacyTokenizer(nlp))
        self.tagger.predict(sentence)
        predictions=sentence.to_dict(tag_type='ner')
        for match in predictions['entities']:
            string=str(match['labels'][0])
            tag=''
            for char in string:
                if char != ' ':
                    tag=tag+char
                else: break
            pattern=nlp.make_doc(match['text'])
            matcher = PhraseMatcher(nlp.vocab)
            matcher.add(tag, None, pattern)
            m = matcher(doc)
            for i in m:
                tok=doc[i[1]]
                if tok.idx==match['start_pos']:
```

```

    span = Span(doc, i[1], i[2], label=i[0])
    doc.ents = list(doc.ents) + [span]
return doc

```

Questo componente andando a sostituire il NER predefinito di SpaCy permette di poter utilizzare le previsioni del modello sviluppato con Flair all'interno dell'ecosistema SpaCy.

Successivamente è stata sviluppata una Prodigy custom recipe (file di configurazione) per poter visualizzare le predizioni del modello Flair attraverso l'efficace interfaccia grafica del tool di annotazione.

La seguente immagine mostra un esempio di referto (testo puramente immaginario) annotato da Prodigy tramite l'utilizzo delle predizioni del modello NER.

The screenshot shows a Prodigy annotation interface. At the top, there is a grid of labels: HCP 1, NAME 2, DATE 3, AGE 4, LOC 5, ADDRESS 6, FAC 7, FAC_NAME 8, REL 9, TAX_CODE 10, PH 11, TIME 12, EMAIL 13, URL 14, PLATE 15, and ZIP 16. Below this, a text document is displayed with various entities highlighted in yellow boxes with blue borders. The text reads:

Al Medico Curante del Sig . Rossi Mario NAME , anni 51 AGE nato a Parma LOC residente a Parma LOC , in Via Garibaldi n 6 ADDRESS , Tel . 3332233111 PH . Paziente con recente ricovero all' ospedale FAC Sacco FAC_NAME di Milano LOC per recidiva di crisi epilettica . Paziente con pregresso ictus talamo - capsulare e temporo - parietale sinistro , diabete mellito tipo II , ipertensione arteriosa . Pregresso ricovero presso la nostra Stroke Unit nel 2007 DATE per ictus ischemico talamo - capsulare sinistro sottoposto a trombolisi . Nel 2008 DATE recidiva di ictus emisferico sn esordito come afasia di Wernicke . Ricovero presso il nostro Reparto (novembre 2013 DATE) per crisi epilettica generalizzata avvenuta con primo episodio la sera del 13/11/13 DATE poco dopo l' addormentamento . In data 12/01/2014 DATE mentre il paziente dormiva ha presentato una nuova crisi morfeica . Conclusioni : Epilessia post - stroke . Visita di controllo fissata per il 16/02/2020 DATE alle ore 16:30 TIME . Cordiali Saluti Dr . L. Verdi HCP

At the bottom, there is a toolbar with four buttons: a green checkmark, a red X, a gray circle with a slash, and a gray left arrow.

Figura 6.1: Esempio di documento annotato da Prodigy con le predizioni del modello

Tutto questo è stato possibile inserendo all'interno della recipe il componente NER custom con al suo interno il modello Flair.

La seguente parte di codice rappresenta una sezione della custom recipe all'interno della quale viene specificata la pipeline del modello e tutti gli argomenti necessari a far partire l'interfaccia grafica.

```
# Recipe decorator with argument annotations: (description, argument type,
# shortcut, type / converter function called on value before it's passed to
# the function). Descriptions are also shown when typing --help.

@prodigy.recipe(
    "ner.make-gold",
    dataset=("The dataset to use", "positional", None, str),
    source=("The source data as a JSONL file", "positional", None, str),
    label=("One or more comma-separated labels", "option", "l",
    split_string),
    exclude=("Names of datasets to exclude", "option", "e", split_string),
)
def ner_make_gold(
    dataset: str,
    source: str,
    label: Optional[List[str]] = None,
    exclude: Optional[List[str]] = None,
):
    """
    Create gold-standard data by correcting a model's predictions manually.

    # Load the spaCy model
    nlp = spacy.load("it_core_news_sm")
    flair_ner = Custom_Ner(nlp, tagger)
    nlp.remove_pipe("ner")
    nlp.add_pipe(flair_ner)

    # Load the stream from a JSONL file and return a generator that yields
    # a dictionary for each example in the data.
    stream = JSONL(source)

    # Tokenize the incoming examples and add a "tokens" property to each
    # example. Also handles pre-defined selected spans. Tokenization allows
    # faster highlighting, because the selection can "snap"
    # to token boundaries.
    stream = add_tokens(nlp, stream)

    # Add the entities predicted by the model to the tasks in the stream
    stream = make_tasks(nlp, stream, label)
```

```

    return {
      "view_id": "ner_manual", # Annotation interface to use
      "dataset": dataset, # Name of dataset to save annotations
      "stream": stream, # Incoming stream of examples
      "exclude": exclude, # List of dataset names to exclude
      "config": { # Additional config settings, mostly for app UI
        "lang": nlp.lang,
        "labels": label, # Selectable label options
      },
    }
}

```

Attraverso la custom recipe è possibile visualizzare le predizioni del modello sui referti direttamente attraverso la grafica di Prodigy. Inoltre è possibile incrementare la grandezza del dataset annotato in modo molto più rapido. Utilizzando la custom recipe e le previsioni del modello l'annotazione di nuovi documenti risulta essere molto più veloce; si passa da una media di circa 40 referti all'ora a circa 170/180 referti all'ora. Le predizioni del modello sono visualizzate direttamente sui referti e l'annotatore non deve fare altro che limitarsi a correggere eventuali errori. Di conseguenza aumentando la grandezza del dataset annotato si può ri-addestrare un nuovo modello che risulterà probabilmente avere prestazioni migliori.

La seguente immagine mostra infatti quanto è migliorato il modello aumentando le dimensioni del dataset annotato e quanto ancora può migliorare.

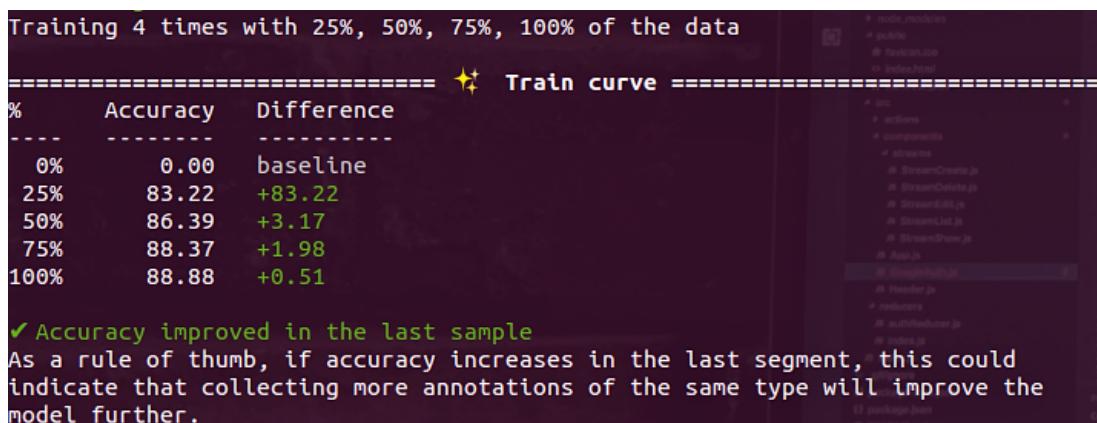


Figura 6.2: Train-curve modello Intermedio

L'immagine appena mostrata rappresenta la funzione train-curve di Prodigy. Come viene scritto nell'immagine, se vi è un incremento dell'accuracy nell'ultimo segmento il modello potrebbe migliorare ancora aumentando la grandezza del dataset annotato.

Capitolo 7

Next steps e conclusioni

7.1 Next steps

- Attraverso la custom recipe sviluppata in Prodigy è possibile incrementare la grandezza del dataset annotato. Si può quindi ri-addestrare un nuovo modello con prestazioni migliori.
- Per migliorare le prestazioni del sistema si possono inserire inoltre dei pattern per le entità che ne permettono lo sviluppo (come ad esempio le email, i numeri di telefono, il CAP e il codice fiscale).
- Una volta sviluppati i precedenti miglioramenti bisognerà poi definire una politica di anonimizzazione o pseudonimizzazione per ciascuna entità.
- Il sistema andrà quindi adattato alle policy di anonimizzazione scelte e poi sarà in grado di deidentificare documenti sanitari di ogni tipo.

Per lo sviluppo degli step appena elencati è necessario collocare il servizio all'interno di un contesto concreto di utilizzo.

Sviluppando quindi i “*Next Step*” e adattando il servizio al caso d'uso in cui verrà effettivamente impiegato, sarà di conseguenza possibile utilizzare questo strumento realmente.

7.2 Conclusioni

La pandemia globale sviluppatasi nel corso dell'anno 2020 ha messo in luce l'importanza della ricerca, che ormai come abbiamo visto, non può più prescindere dal fondamentale apporto dell'Intelligenza Artificiale. L'utilizzo del machine learning in campi come quello medico permette infatti di dare una grossa mano agli operatori sanitari nella ricerca di cure che possono salvare numerose vite umane.

Il libero utilizzo della grande mole di dati in possesso agli enti sanitari non sarà mai possibile fino a quando questi conterranno informazioni riservate e private. Lo sviluppo di servizi di anonimizzazione come quello che è stato proposto in questo elaborato risulta essere quindi più che mai fondamentale.

Tutto questo nella speranza di un futuro nel quale l'uomo e la macchina possano andare a braccetto in un processo di miglioramento della vita di tutta l'umanità e del nostro pianeta.

Bibliografia

- 1) "Codice in materia di protezione dei dati personali" Decreto Legislativo 30 giugno 2003, n. 196
- 2) "General Data Protection Regulation", Regolamento (UE) 2016/679 del Parlamento europeo e del Consiglio del 27 aprile 2016
- 3) "Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule", November 26, 2012
- 4) "Anonimizzazione secondo il WP29 del 2014", Gruppo di lavoro articolo 29 per la protezione dei dati.
- 5) T.M. Mitchell. Machine Learning. McGraw-Hill International Editions. McGraw-Hill, 1997, pp. 81–88.
- 6) "Efficient Estimation of Word Representations in Vector Space" by Mikolov, Tomas; et al. (2013).
- 7) "Distributed representations of words and phrases and their compositionality". by Mikolov, Tomas (2013).
- 8) "GloVe: Global Vectors for Word Representation" by Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
- 9) "Enriching Word Vectors with Subword Information" by Piotr Bojanowski, Edouard Grave, Armand Joulinand, Tomas Mikolov. Facebook AI Research
- 10) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. Google AI Language
- 11) "Contextual String Embeddings for Sequence Labeling" by Alan Akbik, Duncan Blythe, Roland Vollgraf
- 12) MUC-07 Proceedings (Named Entity Tasks)
https://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_toc.html#named

- 13) "May I Check Again? A simple but efficient way to generate and use contextual dictionaries for Named Entity Recognition. Application to French Legal Texts" by Valentin Barriere e Amaury Fouret
- 14) Prodigy, tool di annotazione. <https://prodi.gy/>
- 15) Spacy <https://spacy.io/>
- 16) "Neural Architectures for Named Entity Recognition" by Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer
- 17) Framework SpaCy "*Embed. Encode. Attend. Predict*".
<https://explosion.ai/blog/deep-learning-formula-nlp>
- 18) Reti convoluzionali per l'NLP.
<https://towardsdatascience.com/convolutional-neural-network-in-natural-language-processing-96d67f91275c>
- 19) "Long short-term memory". In: Neural computation 9.8 (1997), pp. 1735–1780. Sepp Hochreiter e Jürgen Schmidhuber.
- 20) "Bidirectional lstm-crf models for sequence tagging" by Zhiheng Huang, Wei Xu, and Kai Yu. 2015
- 21) Progetto anonimizzazione casi legali francesi.
<https://towardsdatascience.com/why-we-switched-from-spacy-to-flair-to-anonymize-french-legal-cases-e7588566825f>
- 22) Progetto anonimizzazione casi legali francesi.
<https://towardsdatascience.com/benchmark-ner-algorithm-d4ab01b2d4c3>
- 23) "Alternative Weighting Schemes for ELMo Embeddings" by Nils Reimers and Iryna Gurevych.
 Ubiquitous Knowledge Processing Lab (UKP-TUDA)
 Department of Computer Science, Technische Universität
- 24) "BioFLAIR: Pretrained Pooled Contextualized Embeddings for Biomedical Sequence Labeling Tasks" by Shreyas Sharma, Ron Daniel, Jr.
- 25) "Frustratingly Easy Meta-Embedding – Computing Meta-Embeddings by Averaging Source Word Embeddings" by Joshua N Coates, Danushka Bollegala