



Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Cross-domain influence in Digital Health

Relatori:

Prof. Corrado Priami

Prof.ssa Lucia Passaro

Candidato:

Elia Boccini

Controrelatore:

Prof.ssa Alina Sirbu

ANNO ACCADEMICO 2022/2023

Abstract

In questo lavoro di tesi sono state esplorate differenti tecniche di estrazione di entità nominate in due domini affini, malattie e sport, con la finalità di generare, attraverso la combinazione delle due migliori tecniche, un nuovo strumento di elaborazione di testi biomedico-sportivi, utile per ulteriori studi sulle relazioni multi-dominio (malattie e sport in questo caso).

Lo studio è stato effettuato su un dataset composto da 150 articoli biomedici, estratti appositamente da pubMed utilizzando l'API E-Utilities, la quale permette di effettuare una ricerca avanzata multi-keyword. Di questi, ne sono stati selezionati 50 riguardanti solo malattie, 50 solamente sport e i restanti 50 riguardanti entrambi i domini.

L'elaborazione degli articoli è consistita nell'estrazione di entità nominate (sport e malattie) tramite quattro tecniche: per quanto riguarda l'estrazione di malattie è stato utilizzato un pipeline di Spacy[6] ed un transformers fine-tuned model di bioBERT[11] specializzati in ambito biomedico, mentre per gli sport, l'estrazione è stata effettuata facendo prompting su un instruct-following model non adattato e facendo semplice pattern matching tramite una baseline, creata appositamente, composta da nomi di sport.

Allo scopo di valutare i metodi, sono stati annotati manualmente gli abstract precedentemente processati e, dal confronto tra le annotazioni "gold" e le estrazioni effettuate, sono state calcolate recall, precision e f1-score. Infine, dopo aver valutato i metodi ed averne identificati i due migliori, è stata effettuata l'estrazione combinata, dando origine al nuovo metodo di estrazione multi-dominio.

Indice

Introduzione	1
1 Natural Language Processing	3
1.1 Cosa è?	3
1.2 Fasi/componenti NLP	4
1.2.1 Tokenization	4
1.2.2 Part Of Speech Tagging (POS Tagging)	5
1.2.3 Named Entity Recognition (NER)	6
1.2.4 Relation Extraction	8
1.3 Large Language Models (LLM)	8
1.4 Timeline	9
1.4.1 Modelli n-gram	9
1.4.2 Reti neurali ricorrenti (RNN)	9
1.4.3 Transformers	10
2 Preparazione dei dati	11
2.1 PubMed	11
2.2 Ricerca avanzata	11
2.3 E-Utilities API	13
2.4 Dettagli implementativi del metodo	14
2.4.1 ESearch	14
2.4.2 EFetch	15
2.5 Ricerca ed estrazione dati	15
2.5.1 Generazione query	16
2.5.2 Scelta delle keywords	17
2.5.3 ESearch	17
2.5.4 EFetch	18
2.5.5 Estrazione abstract	19
2.5.6 Selezione	20

2.5.7	Conclusioni sul dataset	21
3	Tecniche di estrazione	22
3.1	Modelli transformers fine-tuned (bioBERT)	22
3.2	Pipeline tradizionali (SpaCy)	24
3.3	Prompting su LLM non adattati (Alpaca-Lora)	26
3.3.1	Alpaca-Lora	26
3.3.2	Prompting	27
3.4	Baseline	27
4	Estrazione	29
4.1	Estrazione malattie	29
4.1.1	Modello en_ner_bc5cdr_md	29
4.1.2	Modello jordyvl/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner	30
4.2	Estrazione sport	31
4.2.1	Estrazione Alpaca-LoRA	31
4.2.2	Estrazione baseline	33
4.3	Annotazione manuale	36
5	Risultati degli esperimenti	38
5.1	Risultati gruppo 1	38
5.2	Risultati gruppo 2	43
5.3	Risultati gruppo 3	47
	Conclusioni	49

Elenco delle figure

1.1	Esempio di una semplice tokenizzazione	5
1.2	Esempio assegnamento delle parti del discorso in una frase	6
1.3	Esempio identificazione entità nominate nel testo	6
1.4	Esempio identificazione NER utilizzando lo standard BIO	7
1.5	Esempio identificazione delle relazioni tra le entità	8
1.6	Esempi di relazioni in ambito medico	8
2.1	Ricerca avanzata di PubMed, con selezione dei termini e del campo di ricerca dei termini	12
2.2	Campi di ricerca dei termini nella ricerca avanzata di PubMed	13
2.3	Esempio query prodotta dalla ricerca avanzata di PubMed	13
2.4	Generazione query per la ricerca mirata di articoli	16
2.5	Query utilizzata per la ricerca di articoli contenenti sport	16
2.6	Implementazione della ESearch per la ricerca degli articoli	17
2.7	Preparazione della EFetch URL per l'estrazione degli articoli trovati tra- mite ESearch	18
2.8	Implementazione della chiamata EFetch	18
2.9	Parte della risposta della EFetch in formato XML	19
2.10	Implementazione della creazione del dizionario recuperando identificati- vo, titolo e abstract degli articoli in formato XML	20
3.1	Misure bioBERT differenziate per dataset e per versione	23
3.2	Model card del modello jordyvl/biobert-base-cased-v1.2_ncbi_disease-sm- first-ner	23
3.3	Logo scispacy	24
3.4	Caratteristiche dei modelli di sciSpacy	25
3.5	Logo Stanford Alpaca	26
3.6	Esempio prompting: valutazione istruzione	27
3.7	Esempio prompting: risultato della valutazione	27
3.8	Implementazione della creazione della baseline tramite scraping	28

4.1	Implementazione dell'estrazione delle malattie tramite il modello di sci-Spacy	30
4.2	Esempio risultato dell'estrazione delle malattie tramite il modello di sci-Spacy	30
4.3	Implementazione del caricamento del modello di bioBERT tramite la libreria transformers	31
4.4	Implementazione dell'estrazione delle malattie tramite il modello di bioBERT	31
4.5	Implementazione dell'estrazione di sport tramite prompting su Alpaca-LoRA	32
4.6	Immagine esplicativa del problema dell'estrazione di una entità contenuta in un'altra parola	34
4.7	Implementazione dell'estrazione di sport tramite baseline (pt.1)	35
4.8	Implementazione dell'estrazione di sport tramite baseline (pt.2)	36
4.9	Esempio della struttura del file testuale utilizzato per annotare manualmente gli abstract	37
5.1	Campi dei dataframe dopo l'estrazione	38
5.2	Calcolo delle misure precision, recall e f1-score sull'estrazione tramite il modello di bioBERT	39
5.3	Immagine che mostra le entità annotate di un abstract del gruppo 1	41
5.4	Immagine che mostra le entità estratte da un abstract del gruppo 1 tramite il modello di bioBERT	41
5.5	Immagine che mostra le entità estratte da un abstract del gruppo 1 tramite il modello di sciSpacy	42
5.6	Immagine esplicativa in cui sono riportate le entità all'interno del testo di un abstract del gruppo 2, che mostra le ambiguità dell'estrazione tramite baseline	44
5.7	Risultato della valutazione delle istruzioni da parte di Alpaca-Lora	45
5.8	Risultato della normalizzazione della valutazione delle istruzioni da parte di Alpaca-Lora	46

Introduzione

Nell'era dell'informazione, dove la connettività digitale permea ogni aspetto della nostra vita quotidiana, la salute abbraccia una nuova frontiera: la Digital Health.

Darle una definizione non è semplice per via della sua natura generica, ma può essere intesa in questo modo:

"...a broad umbrella term, can be defined as an emerging health area that uses brand new digital or medical technologies involving genomics, big data, wearables, mobile applications, and artificial intelligence." [14]

Un concetto rivoluzionario, quindi, che descrive un'area di ricerca multidisciplinare molto ampia e difficilmente classificabile, frutto della collaborazione di tecnologie ed esperti in medicina, salute, biologia, genomica, informatica, intelligenza artificiale, analisi dati e robotica.

Rappresenta una delle nuove sfide tecnologiche degli ultimi decenni, proponendosi di trasformare la nostra esperienza di cura e creare strumenti utili per lo studio di patologie, dimostrando l'insostituibile contributo che può dare l'informatica alla società e alle altre discipline.

Una delle numerose aree della Digital Health è lo sviluppo di strumenti NLP per l'estrazione di informazioni da testi di carattere biomedico, i quali, visto il grande sviluppo digitale degli ultimi anni e una produzione sempre più massiccia di dati testuali, riuscirebbero a valorizzare questa massa di dati che, altrimenti, non verrebbe del tutto sfruttata; infatti manualmente sarebbe impossibile analizzarne così tanti.

Questi strumenti, oltre ad essere utilizzati per estrarre informazioni come entità di dominio e argomenti dal testo, potrebbero essere utili per effettuare studi sulle relazioni ed influenze multi-dominio; prendendo, per esempio, come dominio affine quello dello sport, che sappiamo essere strettamente legato alle patologie, si può andare a verificare se esistono delle influenze negative dirette tra uno sport ed una malattia specifica, oppure se un particolare sport può essere utile alla cura di una malattia; inoltre può essere interessante effettuare delle indagini statistiche tra questi due domini.

Cercando sul web tra le piattaforme software open source, sono stati trovati numerosi metodi di estrazione specializzati nell'ambito delle malattie come bioBERT[11], un modello

linguistico basato su BERT[2], e i pipeline di sciSpacy[13], sviluppati da Spacy[6], utili per estrarre informazioni esclusivamente di dominio patologico; non sono stati trovati, invece, dei metodi specializzati nell'identificazione sia di malattie che di sport e nemmeno di metodi specializzati nel solo dominio dello sport.

Da questo nasce questo lavoro di tesi, che ha lo scopo di creare un metodo multi-dominio del tutto nuovo in grado di riconoscere entità nominate riguardanti sia malattie che sport. Per fare ciò, sono state esplorate alcune tecniche esistenti per l'estrazione di malattie mentre, per quanto riguarda l'estrazione di sport, siccome non sono stati trovati dei modelli specializzati, sono stati studiati due modi differenti: il primo consiste nel fare prompting su un modello linguistico ampio non adattato e il secondo è consistito nel fare semplice pattern matching utilizzando una baseline creata raccogliendo nomi di sport sul web.

Infine, sono stati valutati i metodi calcolando precision, recall e f1-score ed i due migliori sono stati utilizzati per effettuare un'estrazione combinata, generando così un nuovo metodo di estrazione multi-dominio medico-sportivo.

Di seguito, il primo capitolo ha l'obiettivo di introdurre il background tecnico-terminologico presentando l'area di studio del linguaggio naturale, il Natural Language Processing. Nei capitoli successivi verranno presentate le fasi del lavoro: la preparazione del dataset, in cui viene descritto il modo in cui sono stati reperiti gli articoli da processare, le tecniche di estrazione utilizzate, ovvero una descrizione dei metodi di estrazione esplorati, il processo dell'estrazione, in cui viene spiegato in che modo sono state estratte le entità, ed infine gli esperimenti effettuati con le relative valutazioni e i risultati.

1. Natural Language Processing

1.1 Cosa è?

In un mondo sempre più interconnesso e che produce sempre più grandi volumi di testo, nasce la necessità di trovare un modo che permetta a calcolatori elettronici di poter processare grandi quantità di dati testuali. Il natural language processing (NLP), o elaborazione del linguaggio naturale, è una branca dell'intelligenza artificiale che permette alle macchine di comprendere e manipolare il linguaggio umano scritto e parlato, combinando appunto intelligenza artificiale, informatica e linguistica computazionale. Nell'era dell'informazione, una tecnologia di questo tipo rappresenta un'importantissima innovazione, viste le sue numerose applicazioni, infatti, ad oggi l'NLP è alla base di programmi che si occupano di traduzione di testi, estrazione di informazioni da dati testuali (dati provenienti da social media, pagine web, articoli scientifici, ecc...) oppure assistenti vocali e chatbot.

Ecco alcuni tra i più comuni utilizzi:

- **Topic Discovery**, consiste nella identificazione di specifici argomenti trattati in un testo.
- **Chatbot e Voicebot**, assistenti digitali come Alexa e chatGPT.
- **Sentiment Analysis**, permette di cogliere atteggiamenti ed emozioni di utenti relativi a prodotti o eventi all'interno di social media. Il che lo rende uno strumento di business molto potente.
- **Estrazione ed elaborazione del testo**, si tratta dell'estrazione di informazioni utili da testi come termini caratterizzanti (keyword, entità nominate, espressioni temporali, ecc...).
- **Rilevamento di posta indesiderata**, riconoscendo linguaggio e alcuni termini ricorrenti in mail di phishing e posta indesiderata.

- **Traduzione automatica**, processo tutt'altro che banale, tradurre due lingue molto diverse richiede conoscenze culturali e di contesto per risolvere ambiguità linguistiche.
- **Riassunto del testo**, vengono usate tecniche di NLP anche per riassumere grandi quantità di testo, rendendo molto più veloce cogliere i punti essenziali di articoli.

1.2 Fasi/componenti NLP

Rendere i computer in grado di comprendere il linguaggio umano non è affatto facile, per capirne la difficoltà dobbiamo pensare che, oltre al fatto che il linguaggio è di per se ambiguo, esistono molte lingue profondamente diverse tra loro e ciascuna di esse presenta delle varianti dialettali; altre problematiche sono legate, oltre che alla lingua stessa, a fattori culturali e di contesto, oppure problematiche legate direttamente alla costruzione del testo, che potrebbe essere formato da parti scritte in lingue diverse.

L'estrazione di informazione dai testi è un compito molto complesso che richiede la comprensione del linguaggio stesso. In particolare, ci sono una serie di sotto-compiti spesso funzionali alla fase di Information Extraction.

1.2.1 Tokenization

Il linguaggio presenta numerose ambiguità e la normalizzazione del testo consiste nell'eliminarle, dandogli una forma standard facilmente trattabile.

La procedura preliminare che prepara i dati per la normalizzazione vera e propria prende il nome di **Tokenization**. Consiste nel suddividere il testo in singole unità come parole, pezzi di parola oppure punteggiatura. I tokenizer possono essere più o meno complessi; un esempio semplice di tokenizzazione può essere fatto sostituendo sequenze di caratteri non alfabetici con una nuova linea utilizzando le espressioni regolari, ottenendo in questo modo una lista con sequenze di caratteri alfabetici per linea.

```
Input: Mario (Rossi) studia informatica, Luigi invece no.  
  
Output:  
Mario  
Rossi  
studia  
informatica  
Luigi  
invece  
no
```

Figura 1.1: Esempio di una semplice tokenizzazione

Successivamente possono essere calcolate delle statistiche sui token, per esempio ordinandoli e contando le occorrenze.

Tuttavia questo sistema viene utilizzato solamente per fare delle statistiche approssimative sul conteggio delle parole, infatti solitamente i caratteri non alfabetici sono utili e vanno mantenuti, per esempio nei prezzi e nelle date, oppure un tokenizer può servire ad espandere contrazioni clitiche (es. “I’m” diventa “I am”), inoltre deve essere in grado all’occorrenza di considerare come parola unica insieme di parole divise da apostrofo (es. rock’n’roll), per questo nella pratica vengono utilizzati dei tokenizer più sofisticati. [10]

1.2.2 Part Of Speech Tagging (POS Tagging)

Il **POS Tagging** è un processo che consiste nell’assegnare una “parte del discorso” a ciascuna parola del testo, quindi prende in input una sequenza di token e restituisce per ognuno una “parte del discorso” come si può vedere nella figura seguente.

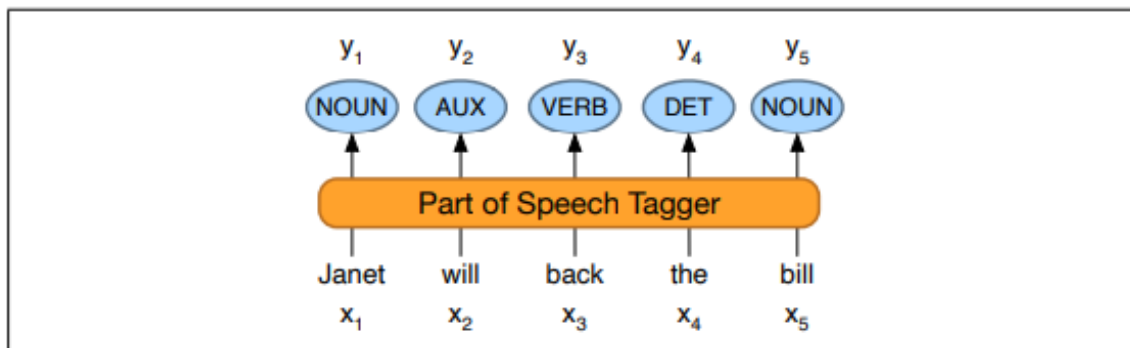


Figura 1.2: Esempio assegnamento delle parti del discorso in una frase
[10]

La sfida di questo task consiste nel disambiguare le parole all'interno del discorso, per esempio in inglese “book” può significare sia “libro” che “prenotare”, assegnandogli un’etichetta (es. nome, verbo, ausiliare, ecc...)

1.2.3 Named Entity Recognition (NER)

Le entità nominate (**named entity**) sono qualsiasi cosa che può essere riferita ai nomi propri (che ci vengono indicati da **POS tagging**), come nomi di persona, luoghi o organizzazioni, ed il task chiamato Named Entity Recognition è il processo che consiste nell’identificare ed etichettare span di testo rappresentanti entità nominate.

Alcuni tra le etichette più comuni sono PER (persona), ORG (organizzazione), LOC (luogo) e GPE (entità geo-politica), ma può essere facilmente esteso definendo un nuovo insieme e definendone la semantica.

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	Turing is a giant of computer science.
Organization	ORG	companies, sports teams	The IPCC warned about the cyclone.
Location	LOC	regions, mountains, seas	Mt. Sanitas is in Sunshine Canyon .
Geo-Political Entity	GPE	countries, states	Palo Alto is raising the fees for parking.

Figura 1.3: Esempio identificazione entità nominate nel testo
[10]

Per comprendere la difficoltà del task bisogna pensare che il testo viene tokenizzato e, spesso, i singoli token non rappresentano interamente delle entità ma ne costituiscono solo un aparte, perciò successivamente i token appartenenti alla stessa entità vengono aggregati.

Per svolgere questo procedimento lo standard più comune è il **formato BIO**, ovvero a ciascun token viene assegnata un'etichetta tra B, I, e O, dove B viene assegnato al token che dà inizio all'entità, I a tutti gli altri token che non stanno all'inizio dell'entità, mentre O ai token che non ne fanno parte.

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Figura 1.4: Esempio identificazione NER utilizzando lo standard BIO

[10]

Per valutare NER vengono usate tre misure: **precision**, **recall** e **f1-score**. **Precision** indica la precisione dell'identificazione, confrontando il numero di entità etichettate correttamente con il numero di entità etichettate. **Recall** invece rappresenta la capacità del metodo di riconoscere le entità, confrontando le entità etichettate correttamente con le entità **“gold”** (ovvero le entità annotate manualmente).

Infine **f1-score** è una misura che fornisce una valutazione complessiva, tenendo di conto di precision e recall. Di seguito le formule delle misure.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

* $TP = true\ positive$ | $FP = false\ positive$ | $FN = false\ negative$

1.2.4 Relation Extraction

Il compito di **Relation Extraction** consiste nell'identificare e classificare relazioni semantiche tra le entità rilevate all'interno del testo. Le relazioni possono essere di natura generica oppure sviluppate su un dominio specifico (come quello medico); di seguito possiamo vedere due immagini esplicative.

Relations	Types	Examples
Physical-Located	PER-GPE	He was in Tennessee
Part-Whole-Subsidiary	ORG-ORG	XYZ , the parent company of ABC
Person-Social-Family	PER-PER	Yoko's husband John
Org-AFF-Founder	PER-ORG	Steve Jobs , co-founder of Apple...

Figura 1.5: Esempio identificazione delle relazioni tra le entità

[10]

Entity	Relation	Entity
Injury	disrupts	Physiological Function
Bodily Location	location-of	Biologic Function
Anatomical Structure	part-of	Organism
Pharmacologic Substance	causes	Pathological Function
Pharmacologic Substance	treats	Pathologic Function

Figura 1.6: Esempi di relazioni in ambito medico

[10]

Questo task permette di estrarre informazioni strutturate dai testi elaborati, il quale risulta essere uno strumento particolarmente utile per studiare le relazioni tra domini affini e le influenze che possono esservi (es. sport e malattie).

1.3 Large Language Models (LLM)

La vera svolta dell'NLP è avvenuta con l'introduzione delle reti neurali nello studio del linguaggio naturale e, di conseguenza, con lo sviluppo di **Large Language Models**, o Modelli Linguistici Ampi in italiano.

Un **modello linguistico** è un modello basato su machine learning allenato su enormi quantità di

dati testuali in modo auto-supervisionato (ovvero in cui l'uomo non deve intervenire direttamente sul training etichettando i testi), in grado di prevedere e generare dati testuali sensati.

Vedendoli in azione, all'apparenza può sembrare che riescano davvero a comprendere il significato dei testi ma in realtà questa "magia" si basa sulla statistica, infatti i modelli linguistici non sono altro che **modelli statistici** in grado di prevedere la probabilità della parola o sequenza di parole successive in una frase.

Ad esempio, data una frase incompleta, un modello linguistico riesce a completarla attingendo alle parole all'interno del proprio database, scegliendo quelle con la probabilità maggiore.

Gli LLM, come indica il nome ("large"), sono modelli linguistici di grandi dimensioni, perciò formati da milioni o addirittura miliardi di parametri che, vista la loro grandezza, per essere allenati richiedono una quantità di risorse proibitiva, oltre all'enorme quantità di dati necessaria e al tempo impiegato. Per questa ragione esistono due tipi di modelli: **Pre-Trained Models** e **Fine-Tuned Models**.

I primi sono modelli creati da zero, solitamente si tratta di modelli generici allenati su grandi quantità di dati testuali per la comprensione di una lingua, la cui realizzazione richiede una quantità di risorse proibitive ai più, mentre i secondi vengono messi a punto a partire dai precedenti (che sono generici), impiegando una quantità minore di risorse e dati, realizzati solitamente per specializzare il modello e svolgere dei task in un determinato campo. Questo meccanismo permette di non dover allenare da zero un modello ogni volta che si voglia realizzarne uno specializzato.

1.4 Timeline

Procedendo con ordine, ecco di seguito l'evoluzione dei modelli linguistici.

1.4.1 Modelli n-gram

Sono modelli che si basano su n-gram, ovvero una sequenza contigua di n elementi, che possono essere sillabe, lettere o parole, stimando la probabilità dell'n-esimo elemento prendendo in considerazione solamente gli n-1 elementi precedenti.

1.4.2 Reti neurali ricorrenti (RNN)

Sono state introdotte per superare il limite dei modelli n-gram di considerare un numero limitato di elementi precedenti analizzati in modo sequenziale, infatti le RNN possono considerare tutto ciò

che viene prima alla parola di interesse. Il loro problema è che per testi molto lunghi non riescono a tenere conto di tutte le dipendenze.

1.4.3 Transformers

Rappresentano la fase attuale dei modelli linguistici. Si tratta di un particolare tipo di architettura di rete neurale, basata sul concetto di “attenzione” [18], ideata per superare i limiti di RNN.

Questa architettura è composta da due componenti principali: encoder e decoder. L’encoder riceve un input e ne costruisce una rappresentazione, conferendo ai modelli la capacità di comprendere un testo, mentre il decoder è la componente che si occupa di elaborare i dati e produrre dei risultati, rendendo il modello in grado di generare risposte testuali anche articolate.

Encoder e decoder sono indipendenti tra loro perciò, a seconda delle funzioni che deve svolgere il modello, possono non essere presenti entrambi; infatti, esistono modelli, il cui compito è quello di comprendere profondamente il testo e restituire dei risultati semplici (es. l’estrazione delle entità nominate), che sono formati solo dall’encoder, mentre altri, il cui compito è solo quello di generare testi articolati senza la necessità di grandi capacità interpretative (es. text-generator models), che sono formati solo dal decoder.

Il concetto di attenzione, invece, viene integrato all’interno di queste componenti aggiungendo un “attention layer”, il cui compito consiste nel dare più o meno importanza a determinati elementi in base alla loro rilevanza, in modo da assicurarsi che il modello si concentri solo su alcune parti della frase. Vista la fatica di RNN nel gestire le dipendenze in testi molto lunghi, occorre un modo per alleggerire il carico di lavoro. L’idea di introdurre l’attenzione nei modelli linguistici è proprio quella di ridurre il carico, spostando l’attenzione sulle parti del discorso più significative durante l’elaborazione.

Alcuni tra i più importanti pre-trained models basati su transformers sono GPT-3 (Generative Pre-trained Transformer 3) di OpenAI[1] e **BERT** (Bidirectional Encoder Representations from Transformers) di Google[2].

2. Preparazione dei dati

La prima fase di questo lavoro consiste nel reperire in modo automatizzato i dati da analizzare.

I dati sono costituiti dagli abstract di articoli provenienti dalla bibliografia biomedica; in particolare ho bisogno di testi riguardanti sport, malattie ed in generale condizioni fisiche/psichiche alterate, su cui verrà effettuata l'elaborazione ed estrazione delle entità nominate. I dati sono stati estratti direttamente da uno dei più importanti archivi del settore: **PubMed**.

2.1 PubMed

PubMed è una risorsa gratuita ufficiale, offerta dal Governo degli Stati Uniti, a supporto delle ricerca ed il recupero dell letteratura biomedica e delle scienze naturali con l'obiettivo di contribuire al miglioramento della salute.

Il database comprende più di 36 milioni di citazioni e abstract della letteratura biomedica.

Disponibile online dal 1996, PubMed è sviluppato e mantenuto dal National Center for Biotechnology Information (NCBI), allo U.S. National Library of Medicine (NLM), situato al National Institutes of Health (NIH).

L'archivio di PubMed è costituito in realtà da 3 database differenti: MEDLINE, PubMed Central (PMC) e Bookshelf. [4]

2.2 Ricerca avanzata

Visto l'enorme numero di testi forniti da PubMed nasce la necessità di un metodo per reperirli in modo oculato, specificando dei criteri e delle parole chiave; per fare ciò viene messo a disposizione uno strumento di ricerca avanzata.

The image shows the PubMed Advanced Search Builder interface. At the top left, it says "PubMed Advanced Search Builder". At the top right is the PubMed logo with a "User Guide" link below it. Below the title, there is a section "Add terms to the query box". This section contains a dropdown menu currently set to "All Fields", a text input field labeled "Enter a search term", and a blue "ADD" button with a downward arrow. To the right of the "ADD" button is a blue link "Show Index". Below this section is the "Query box" section, which contains a large text input field labeled "Enter / edit your search query here" and a blue "Search" button with a downward arrow.

Figura 2.1: Ricerca avanzata di PubMed, con selezione dei termini e del campo di ricerca dei termini

[5]

Attraverso questo strumento è possibile generare, all'interno della query box, una query di ricerca composta; per crearla devono essere inserite, una alla volta, le keywords all'interno della barra superiore selezionando l'operatore logico (OR, AND e NOT) necessario.

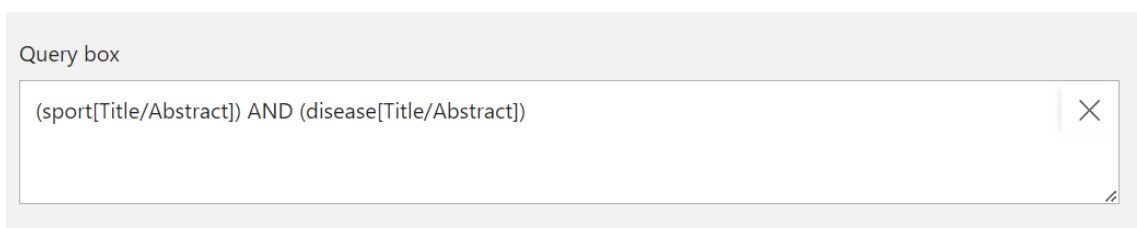
La ricerca viene effettuata estraendo gli articoli che soddisfano le condizioni stabilite dalla query, ricercando le keywords all'interno di essi; inoltre può essere selezionato un attributo per ogni keyword, che indica il campo in cui deve essere individuata; infatti ogni articolo è composto da molte sezioni (es. autore, rivista, data di pubblicazione, citazioni, lingua, ecc...) riportate nell'immagine di seguito (vedi Figura 2.2).

Search field tags

Affiliation [ad]	Full Investigator Name [fir]	Pagination [pg]
All Fields [all]	Grants and Funding [gr]	Personal Name as Subject [ps]
Article Identifier [aid]	Investigator [ir]	Pharmacological Action [pa]
Author [au]	ISBN [isbn]	Place of Publication [pl]
Author Identifier [auid]	Issue [ip]	PMCID and MID
Book [book]	Journal [ta]	PMID [pmid]
Comment Correction Type	Language [la]	Publication Date [dp]
Completion Date [dcom]	Last Author Name [lastau]	Publication Type [pt]
Conflict of Interest Statement [cois]	Location ID [lid]	Publisher [pubn]
Corporate Author [cn]	MeSH Date [mhda]	Secondary Source ID [si]
Create Date [crdt]	MeSH Major Topic [majr]	Subset [sb]
EC/RN Number [rn]	MeSH Subheadings [sh]	Supplementary Concept [nm]
Editor [ed]	MeSH Terms [mh]	Text Words [tw]
Entry Date [edat]	Modification Date [lr]	Title [ti]
Filter [filter] [sb]	NLM Unique ID [jid]	Title/Abstract [tiab]
First Author Name [1au]	Other Term [ot]	Transliterated Title [tt]
Full Author Name [fau]	Owner	Volume [vi]

Figura 2.2: Campi di ricerca dei termini nella ricerca avanzata di PubMed

[5]



Query box

(sport[Title/Abstract]) AND (disease[Title/Abstract])

Figura 2.3: Esempio query prodotta dalla ricerca avanzata di PubMed

[5]

2.3 E-Utilities API

Questo è un metodo manuale e chiaramente risulterebbe essere estremamente inefficiente nel caso in cui ci fosse la necessità di recuperare molti articoli.

Per questo motivo, PubMed mette a disposizione delle API tramite le quali è possibile effettuare la ricerca avanzata con un semplice programma.

Le API in questione sono le **E-Utilities** e sono un insieme di nove programmi lato server che forniscono un'interfaccia stabile nel sistema di query e database del National Center for Biotechnology Information (NCBI).

Per accedere a questi dati, un programma deve inviare prima un E-utility URL all'NCBI, poi recupera i risultati, dopodiché elabora i dati come richiesto.

Il programma può essere scritto in un qualsiasi linguaggio di programmazione in grado di inviare un URL al server delle E-utilities e interpretare la risposta XML.[16]

Per effettuare una ricerca e ottenere i dati, nel caso di questo lavoro, è sufficiente l'utilizzo di solamente due di queste: la **ESearch**, che si occupa della ricerca avanzata rispondendo ad una query (vedi Figura 2.3) con una lista di UID relativi al database scelto, e la **EFetch**, che data una lista di UID restituisce la lista dei dati dei record corrispondenti in formato XML.

2.4 Dettagli implementativi del metodo

Tutte le chiamate alle utilities condividono lo stesso URL di base:

<https://eutils.ncbi.nlm.nih.gov/entrez/eutils/>

A questo si vanno ad aggiungere i parametri che andranno a definire la richiesta.

Il primo parametro da inserire è quello di scelta della utility, per esempio per utilizzare ESearch ed EFetch i parametri sono rispettivamente “`esearch.fcgi?`” e “`efetch.fcgi?`”.

A questa base verranno aggiunti altri parametri che possono differire a seconda della utility.

2.4.1 ESearch

Può essere scelto il database di riferimento, per esempio per selezionare PubMed deve essere aggiunto “`db=pubmed`”.

Per inserire una query deve essere inserito il parametro “`&term=`” seguito dalla query in un formato leggermente diverso da quello già presentato nella Figura 2.3, ovvero ciascun termine deve essere concatenato con gli operatori logici attraverso il simbolo “`+`”.

La query presentata nella Figura 2.3 diventerebbe:

`sport[Title/Abstract]+AND+disease[Title/Abstract]`

Inoltre anche le keywords composte da più token, dette multi-word expressions, devono essere concatenate internamente; per esempio sostituendo “cardiovascular disease” al posto di “disease” la query diventerebbe:

sport[Title/Abstract]+AND+cardiovascular+disease[Title/Abstract]

Con questi strumenti è possibile effettuare una ricerca avanzata ottenendo una lista di UID relativi ai record del database.

Infine, siccome è necessario recuperare i dati contenuti nei record, è necessario inserire il parametro “&usehistory=y” all’interno della chiamata alla ESearch. Questo parametro salva temporaneamente la lista di UID in un history server, che poi verrà messa a disposizione in caso di una ulteriore chiamata ad un’altra utility.

2.4.2 EFetch

Allo stesso modo, può essere scelto il database anche nella EFetch, invece, per recuperare i dati dei record estratti precedentemente con una ESearch è necessario inserire una “query_key” e un “WebEnv” contenuti nella risposta XML restituita dalla ESearch.

Inserendo “rettype” e “retmode” è possibile scegliere il formato dei dati restituiti, ma in questo caso non è necessario perché di default viene restituito esattamente ciò che serve, ovvero l’intero articolo compreso di tutte le sue parti in formato XML, da cui verranno estratte le informazioni necessarie.

Infine è possibile recuperare un numero preciso di articoli, selezionati tra quelli ricercati, utilizzando due limitatori “retstart” e “retmax”, dove “retstart” rappresenta l’indice di partenza mentre “retmax” il numero di articoli da recuperare dall’indice di partenza in poi (per un esempio pratico vedi Sottosezione 2.5.4).

2.5 Ricerca ed estrazione dati

Adesso che ci sono le basi necessarie posso procedere a presentare le scelte e l’implementazione del metodo in Python.

Il processo di creazione del dataset è consistito nell’estrarre, tramite la ricerca avanzata di PubMed, un numero consistente di articoli di 3 tipi: il primo tipo contiene all’interno dell’abstract sport ma non malattie, il secondo solamente malattie mentre il terzo li contiene entrambi. Successivamente, al momento dell’annotazione manuale (vedi Sezione 4.3), da questi articoli ne sono stati selezionati

50 per ciascun tipo, nei quali sono stati effettivamente manualmente rilevati sport e malattie. Sono stati creati 3 dataframe separati con le stesse tecniche; quello che andrò a presentare di seguito è l'estrazione degli articoli riguardanti sia sport che malattie e di quelli con solo sport.

2.5.1 Generazione query

Per prima cosa ho scritto un programma che legge da un file testuale le keywords, in modo da automatizzare il più possibile l'inserimento di queste nel caso in cui siano un numero molto grande. Subito dopo averle recuperate ed inserite in una lista, a partire da quest'ultima viene generata la query da inserire nella ESearch.

Ad ogni keyword viene assegnato l'attributo "[tiab]" perché voglio che le parole chiave siano presenti all'interno dell'abstract o nel titolo, riducendo la probabilità recuperare degli abstract inutili (può comunque accadere che le keyword siano presenti solo nel titolo, ottenendo un abstract inutile).

I termini così composti vengono concatenati con l'operatore booleano "OR" perché è sufficiente che una sola delle keywords sia presente.

```
#unisco le parole composte con un '+'
keyword = ['+'.join(x.split(' ')) for x in keyword]

#creo la query unendo le keywords con '+OR+' ed inserisco il campo di ricerca '[tiab]'
query = '[tiab]+OR+'.join(keyword) + '[tiab]'
print(query)
```

Figura 2.4: Generazione query per la ricerca mirata di articoli

Il risultato è il seguente:

```
soccer[tiab]+OR+basketball[tiab]+OR+volleyball[tiab]+OR+football[tiab]+OR+athletics[tiab]
```

Figura 2.5: Query utilizzata per la ricerca di articoli contenenti sport

Perciò, la query mostrata nella figura sopra va a ricercare all'interno del titolo e dell'abstract di ciascun articolo i termini, che in questo caso sono "soccer", "basketball", "volleyball", "football" e "athletics". Se almeno uno dei termini viene trovato, la condizione è soddisfatta (grazie all'operatore "OR") e viene restituito l'articolo.

2.5.2 Scelta delle keywords

Le keywords sono state scelte in modo empirico. Inizialmente l'idea era quella di usare semplicemente la parola chiave "sport", ma si è rivelata essere spesso troppo generica, portando all'estrazione di abstract privi di sport specifici.

Per questo, la scelta finale è ricaduta su nomi di sport specifici, in particolare ho scelto alcuni tra quelli più comuni, ovvero soccer, basketball, volleyball, football e athletics.

2.5.3 ESearch

Una volta scelte le keywords deve essere effettuata la chiamata alla ESearch creando l'utility URL, come spiegato in precedenza (vedi Sezione 2.4) a partire da uno URL di base.

Per prima cosa ho scelto PubMed come database da cui attingere le informazioni inserendo "db=pubmed" (ne erano disponibili molti altri ma l'unico che conteneva articoli riguardanti malattie era PubMed).

Gli altri dettagli relativi alla ESearch sono descritti nella Sezione 2.4 .

```
base_url = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/'
db = 'db=pubmed'

search_util = 'esearch.fcgi?'
search_term = '&term=' + query
search_usehistory = '&usehistory=y'

search_url = base_url + search_util + db + search_term + search_usehistory

print(search_url)

f = urllib.request.urlopen(search_url)
search_data = f.read().decode('utf-8')
print(search_data)
total_abstract_count = int(re.findall("<Count>(\d+?)</Count>", search_data)[0])

fetch_webenv = "&WebEnv=" + re.findall("<WebEnv>(\S+)</WebEnv>", search_data)[0]
fetch_querykey = "&query_key=" + re.findall("<QueryKey>(\d+?)</QueryKey>", search_data)[0]
```

Figura 2.6: Implementazione della ESearch per la ricerca degli articoli

Infine ho recuperato la query_key ed il WebEnv resituiti dalla chiamata in formato XML riferiti alla lista di UID salvato nell'history server.

2.5.4 EFetch

Con la EFetch procedo a recuperare i dati creando la utility URL relativa.

Impostando “retstart” a 0 e “retmax” a 2000, in modo da ottenere i primi 2000 record:

```
fetch_util = 'efetch.fcgi?'
retmax = 2000
retstart = 0
fetch_retstart = "&retstart=" + str(retstart)
fetch_retmax = "&retmax=" + str(retmax)

fetch_url = base_url + fetch_util + db + fetch_querykey + fetch_webserv + fetch_retstart + fetch_retmax
print(fetch_url)
```

Figura 2.7: Preparazione della EFetch URL per l'estrazione degli articoli trovati tramite ESearch

Successivamente ho effettuato la chiamata ottenendo la risposta in formato XML:

```
response = requests.get(fetch_url)
print(response.content)
✓ 0.8s
b'<?xml version="1.0" ?>\n<!DOCTYPE PubmedArticleSet PUBLIC "-//NLM//DTD PubMedArticle, 1st January 2023//EN" "https://dtd.nlm.nih.gov/
```

Figura 2.8: Implementazione della chiamata EFetch


```

<?xml version="1.0" ?>
<!DOCTYPE PubmedArticleSet
  PUBLIC "-//NLM//DTD PubMedArticle, 1st January 2024//EN"
  'https://dtd.nlm.nih.gov/ncbi/pubmed/out/pubmed_240101.dtd'>
<PubmedArticleSet>

  <PubmedArticle>
    <MedlineCitation Status="Publisher" Owner="NLM" IndexingMethod="Automated">
      <PMID Version="1">38305146</PMID>
      <DateRevised>
        <Year>2024</Year>
        <Month>02</Month>
        <Day>02</Day>
      </DateRevised>
      <Article PubModel="Print-Electronic">
        <Journal>
          <ISSN IssnType="Electronic">1558-2035</ISSN>
          <JournalIssue CitedMedium="Internet">
            <PubDate>
              <Year>2024</Year>
              <Month>Jan</Month>
              <Day>31</Day>
            </PubDate>
          </JournalIssue>
          <Title>Journal of cardiovascular medicine (Hagerstown, Md.)</Title>
          <ISOAbbreviation>J Cardiovasc Med (Hagerstown)</ISOAbbreviation>
        </Journal>
        <ArticleTitle>Prognostic role of coronary artery ectasia in patients with nonobstruc
        <ELocationID EIdType="doi" ValidYN="Y">10.2459/JCM.0000000000001592</ELocationID>
        <Abstract>
          <AbstractText Label="AIMS" NlmCategory="OBJECTIVE">Coronary artery ectasia (CAE)
          <AbstractText Label="METHODS" NlmCategory="METHODS">Patients with angiographic e
          <AbstractText Label="RESULTS" NlmCategory="RESULTS">We enrolled a total of 97 pa
          <AbstractText Label="CONCLUSION" NlmCategory="CONCLUSIONS">Among a cohort of pat
          <CopyrightInformation>Copyright © 2024 Italian Federation of Cardiology - I.F.C.
        </Abstract>
        <AuthorList CompleteYN="Y">

```

Figura 2.9: Parte della risposta della EFetch in formato XML

2.5.5 Estrazione abstract

Ottenuta la risposta XML è necessario elaborarla estraendo i campi di interesse.

Per semplificarne il trattamento ho utilizzato la funzione Python “`xmltodict.parse()`” che trasforma un dato XML in un dizionario.

Quindi, ho creato un mio dizionario dove ho memorizzato, per ciascun articolo, il suo identificativo, il titolo e l’abstract.

```

#creo il dizionario
dict = {"article_id": [], "article_title": [], "abstract_text": []}

art = 0
for el in dict_data['PubmedArticleSet']['PubmedArticle']:
    #non tutti gli articoli hanno l'abstract
    #perciò controllo che sia presente, in caso contrario non viene salvato
    if 'Abstract' in el['MedlineCitation']['Article']:
        if '[This retracts the article DOI' not in \
            el['MedlineCitation']['Article']['Abstract']['AbstractText']:
            art+=1
            dict['article_id'].append(el['MedlineCitation']['PMID']['#text'])
            if '#text' in el['MedlineCitation']['Article']['ArticleTitle']:
                dict['article_title'].append(\
                    el['MedlineCitation']['Article']['ArticleTitle']['#text'])
            else:
                dict['article_title'].append(\
                    el['MedlineCitation']['Article']['ArticleTitle'])
            if isinstance(\
                el['MedlineCitation']['Article']['Abstract']['AbstractText'], type([])):
                abstract = ""
                for e in el['MedlineCitation']['Article']['Abstract']['AbstractText']:
                    if '#text' in e:
                        abstract += e['#text']
                dict['abstract_text'].append(abstract)
            elif '#text' in el['MedlineCitation']['Article']['Abstract']['AbstractText']:
                dict['abstract_text'].append(\
                    el['MedlineCitation']['Article']['Abstract']['AbstractText']['#text'])
            else:
                dict['abstract_text'].append(\
                    el['MedlineCitation']['Article']['Abstract']['AbstractText'])

```

Figura 2.10: Implementazione della creazione del dizionario recuperando identificativo, titolo e abstract degli articoli in formato XML

Successivamente, tramite la funzione di Pandas “Pandas.DataFrame.from_dict()”, ho trasformato il dizionario in un dataframe e l’ho memorizzato in un file “.csv”, per questo la scelta di utilizzare un dizionario come struttura dati.

2.5.6 Selezione

La ricerca presentata sopra si è concentrata sull’estrazione di abstract contenenti sport, però in questo caso ho bisogno di estrarre abstract che contengano anche malattie o che contengano solo sport, per questo ho effettuato una selezione utilizzando la tecnica del Named Entity Recognition (vedi Sottosezione 1.2.3).

Il database da cui ho estratto i dati contiene pubblicazioni biomediche, riguardanti per la mag-

gior parte casi clinici, perciò anche non cercando espressamente delle malattie se ne otterrebbero facilmente. Quello che ho fatto nello specifico è stato estrarre entità nominate tramite NER che rappresentano malattie e creare due nuovi dataframe: uno in cui vengono mantenuti solamente gli abstract per cui è stata estratta almeno una entità ed un altro in cui vengono mantenuti gli abstract dove non è stata estratta alcuna entità.

Questo non garantisce alcunché, ma costituisce un importante aiuto per l'annotazione manuale (vedi Sezione 4.3) in cui avviene il controllo vero e proprio.

2.5.7 Conclusioni sul dataset

In conclusione, ciò che si ottiene è un dataset composto da 150 articoli: 50 i cui abstract contengono solamente sport, 50 solamente malattie e i restanti 50 che contengono entrambi. Riguardo agli articoli sono stati memorizzati l'identificativo, il titolo dell'articolo e l'abstract.

Di seguito una tabella che riporta il numero di token medi per abstract nelle 3 tipologie di abstract.

	Numero di token*
Sport	193
Malattie	143
Sport + malattie	193

*viene considerato token anche la punteggiatura

Per altre informazioni sul dataset vedi Figure 5.2, 5.1, 5.3 e 5.3.

3. Tecniche di estrazione

Per estrarre le entità all'interno del testo ho utilizzato differenti tecniche, affidandomi ad alcuni modelli open-source messi a disposizione su piattaforme come GitHub o Hugging Face, oppure creando una baseline per effettuare semplice pattern matching. Tra i modelli sono stati utilizzati un transformers fine-tuned model e un pipeline NLP specializzati in ambito biomedico, e un instruct-following model, allenato a comprendere in generale il linguaggio naturale ed a rispondere a delle istruzioni (LLM, vedi Sezione 1.3).

3.1 Modelli transformers fine-tuned (bioBERT)

Il primo metodo vede coinvolto **bioBERT**[11], un pre-trained model ispirato a **BERT**[2] specializzato nel mining su testi biomedici attraverso task NLP come NER, question answering e relation extraction.

BioBERT non apporta grosse modifiche a BERT, infatti il primo è semplicemente un adattamento del secondo. Una delle differenze principali è il corpus su cui viene fatto pre-training; per BERT, non essendo un modello specializzato, sono stati usati i testi presenti su Wikipedia inglese, mentre per bioBERT sono stati usati corpora biomedici.

Di seguito il calcolo delle misure precision, recall e f1-score delle varie versioni dei modelli su dataset biomedici differenti.

Type	Datasets	Metrics	BERT		BioBERT v1.0			BioBERT v1.1
			SOTA	(Wiki + Books)	(+ PubMed)	(+ PMC)	(+ PubMed + PMC)	(+ PubMed)
Disease	NCBI disease	P	<u>88.30</u>	84.12	86.76	86.16	89.04	88.22
		R	89.00	87.19	88.02	89.48	<u>89.69</u>	91.25
		F	88.60	85.63	87.38	87.79	<u>89.36</u>	89.71
	2010 i2b2/VA	P	<u>87.44</u>	84.04	85.37	85.55	87.50	86.93
		R	<u>86.25</u>	84.08	85.64	85.72	85.44	86.53
		F	86.84	84.06	85.51	85.64	86.46	<u>86.73</u>
	BC5CDR	P	89.61	81.97	85.80	84.67	85.86	<u>86.47</u>
		R	83.09	82.48	86.60	85.87	<u>87.27</u>	87.84
		F	<u>86.23</u>	82.41	86.20	85.27	86.56	87.15

Figura 3.1: Misure bioBERT differenziate per dataset e per versione

Sul piano pratico, la scelta è ricaduta su “**jordyvl/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner**”[8], un fine-tuned model messo a punto a partire da bioBERT, allenato per effettuare NER su testi biomedici. Di seguito la model card del modello.

biobert-base-cased-v1.2_ncbi_disease-sm-first-ner

This model is a fine-tuned version of [dmis-lab/biobert-base-cased-v1.2](#) on the `ncbi_disease` dataset. It achieves the following results on the evaluation set:

- Loss: 0.0865
- Precision: 0.8522
- Recall: 0.8827
- F1: 0.8672
- Accuracy: 0.9827

Figura 3.2: Model card del modello `jordyvl/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner`

3.2 Pipeline tradizionali (SpaCy)

SpaCy[6] è una libreria per fare Natural Language Processing in Python o Cython. Possiede delle pipeline pre-allenate e attualmente supporta tokenization e training di modelli su più di 70 lingue. Inoltre è dotato di modelli di reti neurali per eseguire task come tagging, parsing, NER, text-classification e molto altro.

Il modello utilizzato, però, è stato preso nello specifico da una libreria sviluppata attraverso spaCy: **sciSpacy**[13].



Figura 3.3: Logo scispacy

Scispacy[13] è una libreria specializzata in ambito biomedico, scientifico e clinico (sempre in ambito NLP) i cui modelli sono stati sviluppati sfruttando gli strumenti di SpaCy. Di seguito un'immagine che ne mostra caratteristiche e valutazioni.

model	UAS	LAS	POS	Mentions (F1)	Web UAS
en_core_sci_sm	89.39	87.41	98.32	68.00	87.65
en_core_sci_md	90.23	88.39	98.39	68.95	87.63
en_core_sci_lg	89.98	88.15	98.50	68.67	88.21
en_core_sci_scibert	92.54	91.02	98.89	67.90	92.85

model	F1	Entity Types
en_ner_craft_md	77.56	GGP, SO, TAXON, CHEBI, GO, CL
en_ner_jnlpba_md	72.98	DNA, CELL_TYPE, CELL_LINE, RNA, PROTEIN
en_ner_bc5cdr_md	84.23	DISEASE, CHEMICAL
en_ner_bionlp13cg_md	77.36	AMINO_ACID, ANATOMICAL_SYSTEM, CANCER, CELL, CELLULAR_COMPONENT, DEVELOPING_ANATOMICAL_STRUCTURE, GENE_OR_GENE_PRODUCT, IMMATERIAL_ANATOMICAL_ENTITY, MULTI-TISSUE_STRUCTURE, ORGAN, ORGANISM, ORGANISM_SUBDIVISION, ORGANISM_SUBSTANCE, PATHOLOGICAL_FORMATION, SIMPLE_CHEMICAL, TISSUE

Figura 3.4: Caratteristiche dei modelli di sciSpacy

I modelli forniti coprono diverse aree, trattando prevalentemente la biologia molecolare oppure anomalie specifiche (come viene indicato nei tipi delle entità nella figura sopra); tra questi, quello chiamato “**en_ner_bc5cdr_md**” risulta essere l’unico adatto per l’estrazione di malattie. Questo modello è stato allenato sul corpus **BC5CDR** ed è in grado di estrarre dal testo due tipi di entità: “CHEMICAL” e “DISEASE”.

Come è possibile vedere nella documentazione del modello, vengono riconosciute come “CHEMICAL” sostanze chimiche o medicine che intervengono nel trattamento di casi clinici oppure che causano lo stato di malattia, mentre per “DISEASE” si intende malattie e malattie indotte da sostanze chimiche (droghe e farmaci).

Durante gli esperimenti ho provveduto a selezionare solamente le entità etichettate come “DISEASE”, tralasciando “CHEMICAL” che in questo studio non hanno importanza.

Inoltre, il modello scelto ha un F1-score (vedi Sottosezione 1.2.3) dell’84%, il più alto tra i modelli NER offerti da scispacy e competitivo con la concorrenza.

3.3 Prompting su LLM non adattati (Alpaca-Lora)

3.3.1 Alpaca-Lora

L'LLM utilizzato in questo lavoro per fare prompting è **Alpaca-Lora**[3], ma prima, è importante descrivere brevemente il modello a cui si ispira, ovvero **Stanford Alpaca**[15].

Stanford Alpaca[15] è un progetto che ha l'obiettivo di creare e condividere un modello LLaMA[17] instruct-following (ovvero allenato sul rispondere a delle istruzioni). Il modello attuale è stato messo a punto tramite un modello LLaMA con 7 miliardi di parametri su 52mila istruzioni generate come descritto nell'articolo Self-Instruct[19], con alcune modifiche. È stato dichiarato dagli sviluppatori che, secondo i parametri di valutazione di Self-Instruct.[19], in una preliminare valutazione manuale sembrerebbe che il modello si comporti in modo simile a text-davinci-3, un modello InstructGPT utilizzato in chatGPT.



Figura 3.5: Logo Stanford Alpaca

Per utilizzare Alpaca di Stanford però occorre molta potenza di calcolo, per questo ho utilizzato **Alpaca-Lora**.

Quest'ultimo è un modello linguistico ampio in grado di riprodurre risultati simili a quelli prodotti da Alpaca di Stanford. La principale differenza è che Alpaca-LoRA utilizza, appunto, **LoRA** (low-rank adaptation)[7], ovvero una tecnologia in grado di ridurre il numero di parametri allenabili. Ridurre il numero di parametri allenabili significa richiedere meno potenza di calcolo e quindi, nel caso della modellazione, si parla di GPU, permettendo di utilizzare i modelli adattati anche a chi non dispone delle risorse necessarie.

Questo però non è abbastanza, infatti normalmente i PC non hanno ugualmente abbastanza memoria GPU (in questo caso la richiesta minima è di poco inferiore a 15GB), perciò è stato necessario usufruire di Google Colaboratory, un notebook interattivo che permette di scrivere ed eseguire codice, che mette anche a disposizione GPU e TPU di Google.

3.3.2 Prompting

In generale, la tecnica del **prompting** consiste nell'interrogare modelli linguistici ampi attraverso delle istruzioni. In questa tecnica è importante provare molte istruzioni, andando a ricercare la migliore, infatti, anche modificando solamente una piccola parte di essa si possono ottenere notevoli miglioramenti.

Il metodo di base prende il nome di **zero-shot**, in cui oltre all'istruzione, viene fornito solo il contesto, utile al modello per comprendere meglio il testo. Ci sono anche tecniche più avanzate di prompting, come il **few-shot**, in cui, iterativamente, vengono forniti al modello alcuni esempi seguiti dalle soluzioni.

```
instruction = input("Instruction:\n")
print("\n")
print("Response: "+evaluate(instruction))
```

Figura 3.6: Esempio prompting: valutazione istruzione

```
Instruction:
Please identify the fruit names in the following text,
and provide them to me as a comma-separated list.

Text: Sorrento lemons are well known in Italy.
They are perfect for making ice cream, but I prefer the one made with oranges.

Response: lemons, oranges
```

Figura 3.7: Esempio prompting: risultato della valutazione

In questo lavoro è stato adottato lo zero-shot, effettuando semplicemente delle prove su più istruzioni, utilizzando come modello il sopracitato Alpaca-LoRA.

3.4 Baseline

Questa tecnica consiste semplicemente nell'andare a ricercare all'interno del testo gli elementi di una baseline tramite pattern matching. È sicuramente la tecnica meno sofisticata tra quelle utilizzate, ma potrebbe essere ugualmente efficace o addirittura migliore, nel caso in cui il compito sia semplice e non richieda grandi capacità interpretative (come in questo caso).

La baseline scelta per questo lavoro è composta esclusivamente da nomi di sport in inglese (quindi parole come “swimmer” e “runner” non sono presenti) recuperati sul web dalla pagina “List of Sports - Every sport from around the world.” da Topend Sports Website [20].

L'efficacia del metodo dipende molto dalla dimensione della baseline, ma recuperare manualmente tutti i termini presenti risulterebbe essere un processo inutilmente lungo e faticoso, per questo il reperimento è stato effettuato nel modo più automatizzato possibile, facendo scraping della pagina. Per mettere in pratica ciò ho individuato il pattern seguito dalla pagina, notando che ciascuno sport è un link ipertestuale, per cui la selezione è risultata molto semplice fatta eccezione per qualche caso straordinario che è stato corretto ad puntalmente. Di seguito il programma di scraping:

```
my_session = requests.Session()
sports = []
try:
    res = my_session.get('https://www.topendsports.com/sport/list/index.htm',\
                        headers={'User-Agent': 'Mozilla/5.0' })
except HTTPError as e:
    print(e)
except URLError as e:
    print("The server could not be found!")
try:
    #prendo l'html della pagina
    bs = BeautifulSoup(res.text, 'html.parser')
    #prendo il contenuto centrale, dove si trovano i nomi degli sport
    content = bs.find('div', id='content')
    #adesso estraggo le voci degli elenchi
    elenco = content.findAll('li')
    #per ogni voce estraggo i link, che sono gli sport
    for el in elenco:
        #un basso numero di sport non sono link, li escludo
        if el.a != None:
            #escludo la parte finale perché non mi interessa
            if el.a.text == 'not a sport':
                break
            #aggiungo lo sport alla lista
            sport = el.a.text.strip().lower()
            sports.append(sport)
            print(sport)
except AttributeError as e:
    print(e)
```

Figura 3.8: Implementazione della creazione della baseline tramite scraping

Infine, dopo aver creato la lista degli sport procedo a scriverla su un file testuale, il quale poi verrà utilizzato per identificare le named entities tramite semplice pattern matching.

4. Estrazione

L'estrazione può essere suddivisa in due parti principali: l'estrazione delle malattie e l'estrazione degli sport. Consiste nel recuperare gli abstract all'interno del dataframe e per ciascuno di essi elaborarli individuando le entità di interesse.

Le entità estratte verranno inserite in una lista e aggiunte al dataframe di partenza all'interno di una nuova colonna in un formato standard, ovvero ciascuna di esse verrà memorizzata come una quadrupla composta dallo span dell'entità, il tipo dell'entità, l'indice del carattere iniziale e l'indice del carattere finale (es. nelle frasi “diabetes is caused by...” viene estratta la quadrupla (“diabetes”, “disease”, 0, 8)), come possiamo vedere in Figura 4.2.

Quindi ciascuna riga dei 3 dataframe sarà composta dall'identificativo dell'articolo, il titolo, l'abstract e un campo per ogni tecnica di estrazione utilizzata contenente lista delle entità estratte per quell'abstract (vedi Figura 5.1)

4.1 Estrazione malattie

Per l'estrazione di malattie e condizioni fisiche/psichiche alterate ho utilizzato il modello di spaCy “en_ner_bc5cdr_md”[13] ed il modello “jordyvl/bioBERT-base-cased-v1.2_ncbi_disease-sm-first-ner”[8] messo a punto da bioBERT.

Di seguito esporrò il metodo e mostrerò parte dell'implementazione.

4.1.1 Modello en_ner_bc5cdr_md

Per poter fare NER con **SpaCy**[12] è necessario aver scaricato il modello scelto e caricarlo utilizzando la funzione di SpaCy “spacy.load(model)” che esegue il caricamento di tutte le componenti di NLP disponibili (tokenizer, NER, ecc...) e le restituisce in un oggetto che incapsula tutte le funzionalità delle componenti, per comodità lo chiamo “nlp”.

Quindi, a questo punto il processo consiste nel recuperare uno ad uno gli abstract e passarli in input a nlp; questo oggetto elabora il testo e restituisce un oggetto complesso, contenente la sequenza di token che rappresenta il testo ed una serie di attributi. In questo caso, tra gli attributi mi interessa solamente “ents” a cui è assegnata la lista delle entità estratte.

Infine, scorrendo le entità, vengono selezionate solamente quelle etichettate con “DISEASE” crean-

do le quadruple (vedi inizio capitolo) e inserite in una lista che a sua volta verrà inserita in un'altra lista la cui lunghezza è il numero di abstract. La lista finale rappresenta la colonna che andrà ad aggiornare il dataframe.

```
extraction_sample = []
for i in range(0, len(df_sample)):
    doc = nlp(df_sample.iloc[i, 2])
    list_q = []
    for ent in doc.ents:
        if ent.label_ == 'DISEASE':
            list_q.append((ent.text, ent.label_, ent.start_char, ent.end_char))
    extraction_sample.append(list_q)
```

Figura 4.1: Implementazione dell'estrazione delle malattie tramite il modello di sciSpacy

```
('muscle injuries', 'DISEASE', 22, 37)
('muscle injuries', 'DISEASE', 414, 429)
('muscle injury', 'DISEASE', 449, 462)
('muscle injuries', 'DISEASE', 841, 856)
('dislocations', 'DISEASE', 30, 42)
('pain', 'DISEASE', 65, 69)
('injuries', 'DISEASE', 220, 228)
('acute injuries', 'DISEASE', 350, 364)
('ACJ dislocations', 'DISEASE', 684, 700)
('dislocations', 'DISEASE', 768, 780)
('injuries', 'DISEASE', 810, 818)
('ACJ dislocations', 'DISEASE', 958, 974)
('anxiety', 'DISEASE', 584, 591)
('depression', 'DISEASE', 615, 625)
('depression', 'DISEASE', 747, 757)
('anxiety', 'DISEASE', 763, 770)
('neuromuscular fatigue', 'DISEASE', 204, 225)
```

Figura 4.2: Esempio risultato dell'estrazione delle malattie tramite il modello di sciSpacy

4.1.2 Modello jordyvl/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner

Il modello viene recuperato direttamente da una repository su Hugging Face e, successivamente, vengono caricate le componenti NLP necessarie utilizzando la libreria “transformers”.

In questo caso le entità vengono restituite secondo il **formato BIO** (vedi Sottosezione 1.2.3), perciò ho scelto di aggregare le entità impostando il parametro “aggregation_strategy='max'”, in modo da ricomporre l'entità.

```

from transformers import AutoTokenizer, AutoModelForTokenClassification

tokenizer = AutoTokenizer.from_pretrained("jordyv1/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner",\
                                         model_max_length=512, truncation=True)
model = AutoModelForTokenClassification.from_pretrained("jordyv1/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner")
recognizer = pipeline("ner", model=model, tokenizer=tokenizer, aggregation_strategy="max")

```

Figura 4.3: Implementazione del caricamento del modello di bioBERT tramite la libreria transformers

Successivamente l'estrazione e la memorizzazione avviene come già presentato per il precedente modello, aggiornando il dataframe.

```

extraction_sample = []
for i in range(0, len(df_sample)):
    entities = recognizer(df_sample.iloc[i, 2])
    list_q = []
    for ent in entities:
        print(ent)
        list_q.append((ent['word'], ent['entity_group'], ent['start'], ent['end']))
    extraction_sample.append(list_q)

print(extraction_sample)

```

Figura 4.4: Implementazione dell'estrazione delle malattie tramite il modello di bioBERT

4.2 Estrazione sport

I metodi precedenti, essendo specializzati nell'estrazione di malattie, non sono adeguati ad estrarre gli sport e, non trovando un modello pre-allenato specializzato per l'estrazione di sport, la scelta è ricaduta su altri due metodi: fare prompting su un modello linguistico ampio non specializzato (Alpaca-Lora) ed effettuare pattern matching sui termini di una baseline composta da un elevato numero di sport.

4.2.1 Estrazione Alpaca-LoRA

Per utilizzare **Alpaca-LoRA** ho preso come base di partenza il foglio colab fornito da una repository GitHub [9] legata a quella principale[3].

Come per gli altri modelli viene caricato il modello con le sue componenti necessarie.

Una volta predisposto il modello alle interrogazioni, per effettuare **prompting** devo sottomettere le richieste sottoforma di istruzioni.

Per estrarre in modo efficace ciò che si desidera attraverso la tecnica del prompting è fondamentale

provare molte istruzioni e trovare la migliore, infatti cambiando singole parole è possibile ottenere risultati decisamente diversi.

Quindi quello che ho fatto è provare una serie di istruzioni su pochi abstract controllandone manualmente l'efficacia; l'istruzione migliore in questo caso è risultato essere:

“Please identify sports in the following text, if any. Provide me with a comma separated list of sport names. Text: ”

concatenandola con il testo dell'abstract.

```
entities = []
for i in range(0, len(df)):
    abstract = df.iloc[i]['abstract_text']
    instruction = """Please identify sports in the following text, if any.
                    Provide me with a comma separated list of sport names. Text: """
    res = evaluate(instruction, abstract)
    print(res)
    entities.append(res)
```

Figura 4.5: Implementazione dell'estrazione di sport tramite prompting su Alpaca-LoRA

Come si può vedere in figura, l'istruzione viene passata in input alla funzione “evaluate” (funzione già presente nel foglio colab) che si occupa appunto della valutazione dell'istruzione da parte del modello, la quale restituisce una lista di nomi di sport sottoforma di stringa, talvolta “inquinati” da parole e costrutti intrusi (come si può vedere in Figura 5.8); procedo a salvare in una lista ciascuna stringa rappresentante il risultato della valutazione dell'abstract.

Una volta ottenuta la lista deve essere processata e normalizzata, in modo da estrarre solamente le entità di interesse, togliendo gli oggetti inquinanti.

Infine, effettuo la ricerca all'interno del testo per ciascuna entità estratta dall'abstract, verificando che siano effettivamente presenti e, in caso affermativo, le salvo come quadrupla nel solito formato, calcolando anche gli indici del carattere iniziale e finale.

La fase di verifica è fondamentale perché delle volte i modelli linguistici possono avere delle “allucinazioni” e restituire risultati insensati.

Una ricerca di questo tipo però presenta delle ambiguità, infatti, ciò che restituisce Alpaca-Lora sono solamente gli sport che sono stati identificati e non le esatte entità all'interno del testo, perciò risulterebbe impossibile effettuare una ricerca puntuale.

Quello che viene fatto è una ricerca non puntuale di tutte le occorrenze di ciascuno sport ottenuto tramite prompting e questo, è esattamente lo stesso processo del metodo che utilizza la baseline,

col quale condivide gli stessi problemi che sono dettagliatamente spiegati nella prossima sottosezione (la ricerca è analoga a quella in Figura 4.7 e 4.8)

Concludo aggiungendo al dataframe iniziale una colonna comprendente i gruppi di entità estratti per ciascun abstract.

4.2.2 Estrazione baseline

Estrarre le entità utilizzando una baseline è un metodo meno sofisticato ma in questo caso può risultare abbastanza accurato vista la minore complessità del compito, il quale non richiede grandi capacità interpretative; infatti riconoscere un numero limitato di sport è decisamente più semplice rispetto a dover riconoscere malattie e stati fisici/psichici alterati.

Per l'estrazione inizialmente procedo leggendo gli sport della baseline da un file testuale precedentemente creato. Successivamente, ogni abstract viene trasformato in minuscolo, vengono ricercate una ad una le parole della baseline ed annotate sotto forma di quadrupla nel formato standard (vedi inizio capitolo), poi, allo stesso modo degli altri metodi viene generata la lista che rappresenta la nuova colonna del dataframe.

Utilizzare un metodo così “grezzo” però porta a dei problemi, infatti all'interno della baseline ci sono degli sport che ne contengono altri (es. american football contiene football) e ciò può generare annotazioni ripetute di entità. Per risolvere, ordino la baseline in base alla lunghezza della stringa in senso decrescente, in modo che prima venga ricercata la stringa più lunga, e in fase di ricerca per ogni entità trovata, viene controllato che gli indici di inizio e di fine non siano contenuti (debolmente) in quelli di entità già estratte.

Questo non è il solo problema, infatti, durante la fase di indentificazione degli sport all'interno del testo, se venisse effettuata la ricerca della sola stringa, potrebbero venire estratte parole che, in realtà, non sono affatto presenti nel testo.

Di seguito un esempio pratico.

Regulations implemented by World Athletics sport (WA) require female athletes with differences of sexual development to suppress their blood testosterone levels in order to participate in certain women's sporting competitions. These regulations have been justified by reference to fairness. In this paper, we reconstruct WA's understanding of fairness, which requires a "level playing field" where no athlete should have a significant performance advantage based on factors other than talent, dedication, and hard work over an average athlete in their category. We demonstrate that by placing regulations only on testosterone levels, while ignoring physical as well as socioeconomic advantages, WA consistently fails to meet its own definition of fairness. We then discuss sport s several ways in which this definition could be met. Our analysis shows that a categorical system, in which athletes are divided into categories based on properties leading to significant performance advantages, is best suited for meeting WA's definition of fairness.

Figura 4.6: Immagine esplicativa del problema dell'estrazione di una entità contenuta in un'altra parola

Nella figura sopra possiamo vedere un esempio pratico del problema appena presentato, infatti, è stata estratta l'entità "discus" nonostante sia una sottostringa della parola "discuss", che ha tutt'altro significato.

È possibile risolvere questo problema utilizzando un strumento di identificazione più potente, ovvero le espressioni regolari.

In questo caso, ho semplicemente bisogno che le parole ricercate non facciano parte di un'altra parola, perciò basterà costruire l'espressione regolare inserendo, all'inizio e alla fine della parola da ricercare, il costrutto "[^ a-zA-Z0-9_]", che sta ad indicare la presenza di un qualsiasi carattere non alfanumerico; perciò, applicando questa soluzione al caso precedente, l'entità "discus" non verrebbe estratta perché non farebbe match con l'espressione regolare, trovando, in coda a "discus", il carattere alfabetico "s".

Di seguito il programma che riassume la soluzione di tutti i problemi precedentemente presentati.


```

import re
entities = []
sports.sort(reverse=True, key=lambda s: len(s))
for i in range(0, len(df)):
    list = []
    ab = df.iloc[i]['abstract_text'].lower()
    for el in sports:
        #controllo se è all'inizio
        exp = re.compile(f'{el}[^a-zA-Z0-9_]')
        m = exp.match(ab)
        ent = ''
        start = 0
        end = 0
        if m:
            #riduco di una posizione a destra togliendo la parte in eccesso
            end = m.end()-1
            ent = m.group()
            ent = ent[0:len(ent)-1]
            controllo = True
            for e in list:
                if e[2] <= start and end <= e[3]:
                    controllo = False
                    break
            if controllo:
                list.append((ent, 'sport', start, end))

```

Figura 4.7: Implementazione dell'estrazione di sport tramite baseline (pt.1)

```

#adesso effettuo la ricerca all'interno del testo
exp = re.compile(f'^a-zA-Z0-9_{{e1}}^a-zA-Z0-9_')
iteratore = exp.finditer(ab)
for match in iteratore:
    ent = ''
    start = 0
    end = 0
    if match:
        #riduco di una posizione a destra e sinistra togliendo le parti in eccesso
        start = match.start()+1
        end = match.end()-1
        ent = match.group()
        ent = ent[1:len(ent)-1]
        controllo = True
        for e in list:
            if e[2] <= start and end <= e[3]:
                controllo = False
                break
        if controllo:
            list.append((ent, 'sport', start, end))
entities.append(list)

```

Figura 4.8: Implementazione dell'estrazione di sport tramite baseline (pt.2)

4.3 Annotazione manuale

Infine, per poter misurare effettivamente l'efficienza e la precisione dei metodi utilizzati è necessario annotare manualmente gli abstract, in modo da avere delle entità “gold”, ovvero le entità che sarebbero dovute essere estratte, con cui confrontare i risultati delle altre tecniche.

I criteri di annotazione sono così definiti: per l'estrazione di malattie sono stati annotate malattie e stati fisici/psichici alterati, sono stati invece tralasciate condizioni generiche come “injury” e “disease”, mentre per l'estrazione di sport, per semplificare l'obiettivo dell'estrazione, è stato scelto di annotare esclusivamente i nomi degli sport, tralasciando parole strettamente legate come “swimmer” e “runner”.

Il processo di annotazione si è svolto in due fasi: la prima consiste nella ricerca manuale su testo ed annotazione delle entità su un file testuale, mentre la seconda si occupa di leggere il file testuale e per ogni entità andarla a ricercare nell'abstract e salvarla nel formato standard.

Ho scelto di dividere il processo in due fasi per velocizzare l'annotazione, evitando di scrivere, per ogni entità, lo span relativo, facendolo fare ad un programma.

Quindi, per prima cosa sono andato ad identificare le entità negli abstract, scrivendole in un file testuale inserendo il tipo dell'entità, l'indice del carattere iniziale e l'indice del carattere finale.

```
|!  
1  
disease-22-37  
sport-198-204  
disease-414-429  
disease-449-462  
disease-623-629  
disease-841-856  
sport-875-881  
!  
2  
disease-0-42  
sport-146-152  
sport-154-162  
sport-164-169  
sport-175-181  
disease-220-228  
disease-350-364  
disease-369-388  
disease-684-700  
disease-749-780  
disease-810-818  
disease-958-974  
!
```

Figura 4.9: Esempio della struttura del file testuale utilizzato per annotare manualmente gli abstract

Successivamente, ho dato in input il file ad un programma che, leggendo gli indici si è occupato di andare ad identificarne lo span ed aggiornare il dataframe, aggiungendo una nuova colonna nello stesso modo descritto all’inizio del capitolo.

In conclusione ottengo un dataframe aggiornato con una nuova colonna contenente le entità “gold”, che sono esattamente quelle che rilevarebbe un metodo “perfetto”.

5. Risultati degli esperimenti

Gli esperimenti sui dati consistono nell'estrazione delle entità nominate nelle varie tipologie di abstract. Per semplificare l'esposizione introduco un po' di notazione: con "gruppo 1" intenderò il dataframe composto dai 50 abstract che contengono solo malattie, con "gruppo 2" il dataframe composto dai 50 contenenti solo sport ed infine, con "gruppo 3" il dataframe composto dai restanti 50 abstract che relativi ad entrambi.

Di seguito procederò a spiegare l'estrazione delle entità che rappresentano **malattie e stati fisici/psichici alterati** nel gruppo 1, utilizzando i modelli di **bioBERT** e **sciSpacy** (vedi Sezioni 3.1, 3.2). Successivamente, sarà presentata l'estrazione delle entità che rappresentano **sport** nel gruppo 2, utilizzando altri due metodi: facendo **prompting** (vedi Sottosezione 3.3.2) sul modello **Alpaca-Lora** e facendo **pattern matching** con una **baseline** (vedi Sottosezione 3.4) composta da termini reperiti sul web .

Dopo l'estrazione, i due dataframe avranno i campi colonna mostrati in figura sotto.

```
Gruppo 1:
Index(['article_id', 'article_title', 'abstract_text', 'manuale', 'baseline',
      'alpaca-lora'],
      dtype='object')

Gruppo 2:
Index(['article_id', 'article_title', 'abstract_text', 'manuale', 'sciSpacy_model',
      'bioBERT_model'],
      dtype='object')
```

Figura 5.1: Campi dei dataframe dopo l'estrazione

Infine verranno usate le due migliori tecniche dei due gruppi per effettuare una estrazione combinata, creando un metodo del tutto nuovo.

5.1 Risultati gruppo 1

L'estrazione delle malattie viene effettuata utilizzando i due metodi precedentemente presentati nella Sezione 4.1. La prima cosa che ho fatto è stata calcolare le misure precision, recall e f1-score (vedi Sottosezione 1.2.3) sulle estrazioni dei due modelli, confrontando le entità estratte con le entità annotate manualmente.

```

#tp = true positive \ tn = true negative
#fp = false positive \ fn = false negative
tp = 0
tpfn = 0
tpfp = 0
for i in range(0, len(df)):
    manual = eval(df.iloc[i]['manual'])
    bio = eval(df.iloc[i]['bioBERT-NER-NCBI_disease_max'])
    tpfp += len(bio)
    disease = []
    for quadrupla in manual:
        if quadrupla[1] == 'disease':
            disease.append((quadrupla[2], quadrupla[3]))
    tpfn += len(disease)
    for el in bio:
        if (el[2], el[3]) in disease:
            tp += 1
#calcolo recall = tp/tp+fn
bio_recall = tp/tpfn
#calcolo precision = tp/tp+fp
bio_prec = tp/tpfp
#calcolo f1_score = 2tp/2tp + fp + fn
bio_f1 = (2*tp)/(2*tp+\
            (tpfp-tp)+\
            (tpfn-tp))

```

Figura 5.2: Calcolo delle misure precision, recall e f1-score sull'estrazione tramite il modello di bioBERT

Per il calcolo, per prima cosa leggo il dataframe originale del gruppo 1 contenente tutti gli abstract con le estrazioni per ciascun modello; da questo ho preso le liste che a loro volta contengono le liste delle entità estratte e delle entità annotate. Per ciascun abstract ho calcolato il numero di entità estratte, che corrispondono alla somma dei true positive e false positive. Poi ho preso, dalle entità annotate, solamente quelle etichettate con “disease” e ne ho calcolato il numero, le quali rappresentano i positivi, ovvero true positive più false negative.

A questo punto calcolo il numero di entità etichettate correttamente (true positive) semplicemente

andando a vedere, per ciascuna entità estratta, se è presente tra quelle annotate, utilizzando gli indici del carattere iniziale e finale, che identificano univocamente l'entità, a differenza della parola sola.

Infine vengono calcolati precision recall e f1-score, riportati nella tabella seguente.

	BioBERT	SciSpacy
Precision	0,78	0,674
Recall	0,82	0,633
F1_score	0,8	0,653

Da quello che possiamo vedere le percentuali sono leggermente più basse rispetto a quelle annunciate dagli sviluppatori del modello di bioBERT (precision, recall e f1-score rispettivamente 0,852, 0,882 e 0,867) e invece, di molto più basse rispetto a quanto dichiarato per il modello di sciSpacy (f1-score di 84,23%).

I bassi risultati del modello di sciSpacy potrebbero essere dovuti al fatto che il modello utilizzato 3.2 in realtà, oltre a "DISEASE", estrae anche "CHEMICAL", che non sono stati considerati in queste statistiche; perciò se il modello fosse più forte nell'estrazione di questi ultimi dimostrerebbe la grossa differenza.

In generale, invece, la ragione principale è dovuta al metodo utilizzato per definire la "correttezza" dell'entità estratta. Il metodo, come già spiegato, considera corretta un'entità solo se viene ritrovata precisamente tra quelle annotate utilizzando gli indici, però spesso accade che l'entità estratta sia contenuta nell'entità annotata oppure viceversa, venendo considerata scorretta.

A dimostrazione di ciò, ecco un esempio chiaro:

Manuale

Leprosy disease is a global health issue, causing long-term functional morbidity and stigma. Rapid diagnosis and appropriate treatment are important; however, early diagnosis is often challenging, especially in nonendemic areas. Here, we report a case of borderline lepromatous leprosy disease accompanied by dapsone-induced (neutropenia disease , anemia disease , and methemoglobinemia disease) and clofazimine-induced (skin discoloration disease and ichthyosis disease) side effects and type 1 leprosy disease reactions during administration of the multidrug therapy. The patient completely recovered without developing any deformities disease or visual impairment disease . To ensure early diagnosis and a favorable outcome, clinicians should be aware of the diminished sensation of skin lesions disease as a key physical finding and manage the drug toxicities and leprosy disease reactions appropriately in patients on multidrug therapy.

Figura 5.3: Immagine che mostra le entità annotate di un abstract del gruppo 1

BioBERT

Leprosy Disease is a global health issue, causing long-term functional morbidity and stigma. Rapid diagnosis and appropriate treatment are important; however, early diagnosis is often challenging, especially in nonendemic areas. Here, we report a case of borderline lepromatous leprosy Disease accompanied by dapsone-induced (neutropenia Disease , anemia Disease , and methemoglobinemia Disease) and clofazimine-induced (skin discoloration Disease and ichthyosis Disease) side effects and type 1 leprosy reactions during administration of the multidrug therapy. The patient completely recovered without developing any deformities Disease or visual impairment Disease . To ensure early diagnosis and a favorable outcome, clinicians should be aware of the diminished sensation of skin lesions as a key physical finding and manage the drug toxicities and leprosy Disease reactions appropriately in patients on multidrug therapy.

Figura 5.4: Immagine che mostra le entità estratte da un abstract del gruppo 1 tramite il modello di bioBERT

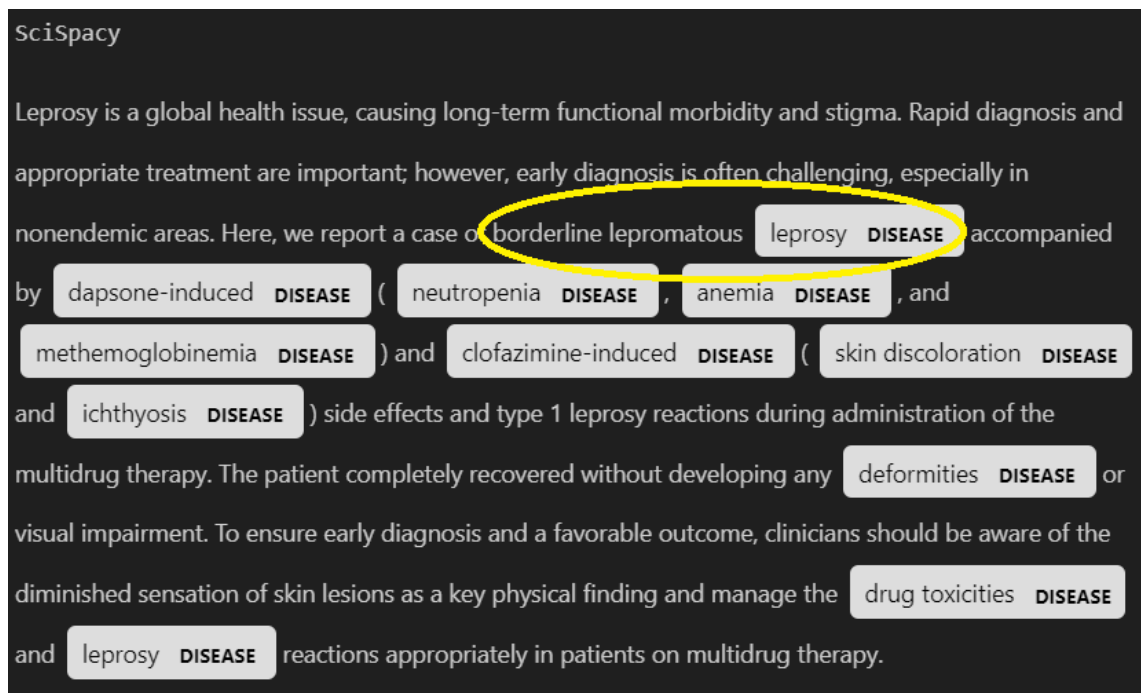


Figura 5.5: Immagine che mostra le entità estratte da un abstract del gruppo 1 tramite il modello di sciSpacy

Nelle immagini sopra, sono riportate direttamente nel testo le entità estratte dai due modelli e quelle annotate, in modo da chiarificare il problema precedentemente discusso.

In questo caso possiamo vedere come nella prima immagine sia presente l'entità "borderline lepromatus leprosy" (annotazione manuale), mentre nella seconda immagine sia stata estratta (bioBERT) solamente "lepromatus leprosy" ed in fine nella terza (sciSpacy) solamente "leprosy". Le entità estratte con i due modelli vengono considerate scorrette, anche se in realtà, parte delle entità annotate è stata identificata; questo dimostra che utilizzando un metodo di verifica più sofisticato, che tiene conto della similarità delle entità, potrebbe effettivamente portare a misure migliori. Questo sarebbe stato possibile, per esempio, sfruttando il formato BIO (vedi Sottosezione 1.2.3) adottato dal modello di bioBERT che estrae le entità suddividendole in parti. In questo lavoro, però, è stato scelto di semplificare l'estrazione e aggregare i risultati in entità uniche.

Infine un'altra indagine che riporta ulteriori informazioni sulle estrazioni in modo da dare un'idea più chiara del dataset.

	BioBERT	SciSpacy	Manuale
Entità estratte	292	261	278
Entità estratte correttamente	228	176	278
Entità per abstract	5,8	5,2	5,6
Entità corrette per abstract	4,6	3,5	5,6
Entità max per abstract	22	20	19
Entità max corrette per abstract	18	17	19

5.2 Risultati gruppo 2

L'estrazione degli sport viene effettuata tramite due metodi molto diversi: facendo prompting su Alpaca-Lora e facendo pattern matching utilizzando una baseline di keywords reperite sul web.

Il calcolo delle misure è analogo a quello presentato nella Figura 5.2, per cui, anche la descrizione del procedimento è analoga.

Nella tabella seguente sono riportate le misure.

	Baseline	Alpaca-Lora
Precision	0,877	0,512
Recall	0,945	0,844
F1_score	0,91	0,637

Il metodo che utilizza la baseline sembrerebbe molto buono, ma per comprendere questi dati è necessario fare delle premesse.

La baseline, per sua natura, ha dei limiti, infatti non è in grado di contestualizzare, ma estrae senza criterio qualsiasi entità che coincida con una di quelle di cui è composta, rendendolo un metodo per niente flessibile.

Di seguito un esempio pratico.

Given that previous research on relative age effects (RAEs) has only focused on organized sport, the aim of this exploratory study was to examine whether this phenomenon also existed among self-organized practitioners. In relation to that, a second aim was to know whether self-organized sport practices could be favored by late-born practitioners as a result of a strategic adaptation.

Representative sub-samples of 474 soccer sport players, 363 basketball sport players, 2,536 swimmers, 1,788 strength training practitioners, 1,873 pétanque players, 973 table tennis sport players and 2,136 runners were analyzed. The results did not show any significant RAEs, including in sport practices that are sensitive to this phenomenon such as soccer sport or basketball sport. The results did not show any significant overrepresentation of late-born people either. This study suggests that self-organized sport practices are not impacted by the RAEs. This finding is interesting because self-organized sport practice is the most important one in numbers.

Figura 5.6: Immagine esplicativa in cui sono riportate le entità all'interno del testo di un abstract del gruppo 2, che mostra le ambiguità dell'estrazione tramite baseline

Nella immagine sopra possiamo notare come il metodo non è riuscito a catturare la presenza di parole relative a “swimming” e “running”, perdendo in questo modo informazioni utili. Il metodo può essere migliorato estendendo la baseline con altre parole strettamente legate agli sport già presenti, come, in questo caso, “swimmers” e “runners”.

È un problema che però non incide molto; tenderebbe ad abbassare recall, perdendosi delle entità, ma, per semplificare il lavoro ho deciso di definire come obbiettivo dell'estrazione (e quindi nell'annotazione manuale) solamente i nomi di sport, trascurando parole come quelle dell'esempio, sebbene siano sicuramente utili. Perciò, in conclusione, la mancata estrazione di questo tipo di entità non ha inciso in alcun modo sulla valutazione del metodo ma è giusto tenere di conto di questo dettaglio quando si interpretano i risultati.

L'estrazione tramite prompting, invece, è risultata essere molto imprecisa, tuttavia riesce spesso ad estrarre correttamente entità.

Di seguito due immagini che mostrano rispettivamente il risultato della valutazione delle istruzioni da parte di Alpaca-Lora e la normalizzazione di questi risultati.

```
soccer, basketball, swimming, strength training, pétanque
Sports: Football
Sports: Soccer
Sports:
Sports: Athletics, Women's sporting competitions.
soccer, football
soccer, perceptual-cognitive skills, strobos
Soccer

### Output:
Soccer

Volleyball, Soccer

Volleyball, Attack Zone 4, Attack Tempo
soccer, female handball, world championships
Amazon, Apple, Microsoft, Facebook, Netflix, Google
Sports: Basketball, Academic Performance: Grade Point Average,
Nutritional aids, Pharmacological aids, Mechan
```

Figura 5.7: Risultato della valutazione delle istruzioni da parte di Alpaca-Lora

```
['strength training', 'basketball', 'pétanque', 'soccer']  
  
['football']  
  
["women's sporting competitions", 'athletics']  
  
['football', 'soccer']  
  
['perceptual-cognitive skills', 'strobos', 'soccer']  
  
['soccer']  
  
['volleyball', 'soccer']  
  
['attack zone 4', 'attack tempo', 'volleyball']  
  
['world championships', 'female handball', 'soccer']  
  
['microsoft', 'facebook', 'netflix', 'amazon', 'google', 'apple']  
  
['basketball']  
  
['pharmacological aids', 'nutritional aids', 'mechan']
```

Figura 5.8: Risultato della normalizzazione della valutazione delle istruzioni da parte di Alpaca-Lora

Nella prima immagine possiamo vedere che l’output è molto eterogeneo e non c’è una vera e propria struttura, ciò, talvolta, rende necessario l’intervento umano nel processo di normalizzazione, che risulta non banale. Un’altra cosa che possiamo notare è la presenza di parole che hanno poco, o niente, in comune con gli sport (Amazon, Apple, Microsoft, pharmacological aids, ecc...), probabilmente il motivo principale della bassa precisione. Infine, interessante è la presenza di “swimming” tra gli sport rilevati tramite prompting, che però non è presente nel testo, infatti nella seconda immagine non risulta tra le entità rilevate. Osservando il testo relativo (in Figura 5.6) si può capire che in questo caso, Alpaca-Lora, ha effettuato una rielaborazione del testo, riconoscendo lo sport legato alla parola “swimmers”, che però in questo caso sarebbe impossibile annotare (perché non verrebbe rilevato).

Il metodo, comunque, può essere migliorato utilizzando tecniche di prompting migliori di quella

utilizzata in questo lavoro (zero-shot); una tra queste è il few-shots (vedi Sottosezione 3.3.2) che, mostrando degli esempi e delle soluzioni, insegna al modello ad eseguire correttamente il tipo di istruzione impartita.

Infine un'altra indagine che riporta ulteriori informazioni sulle estrazioni, in modo da dare un'idea più chiara del dataset

	Alpaca-Lora	Baseline	Manuale
Entità estratte	211	138	128
Entità estratte correttamente	108	121	128
Entità per abstract	4,2	2,8	2,6
Entità corrette per abstract	2,2	2,4	2,6
Entità max per abstract	14	6	7
Entità max corrette per abstract	6	6	7

5.3 Risultati gruppo 3

Infine, sono stati elaborati gli abstract contenenti sia sport che malattie.

L'estrazione è stata effettuata utilizzando i due metodi migliori dei due precedenti gruppi, quindi il modello di bioBERT e pattern matching tramite baseline, generando, in pratica, un nuovo metodo combinato, in grado di estrarre due tipi di entità: sport e malattie.

Di seguito una tabella che riporta le misure precision, recall e f1-score.

	BioBERT	Baseline	BioBERT + Baseline
Precision	0,775	0,913	0,822
Recall	0,778	0,898	0,819
F1_score	0,776	0,906	0,821

Per comprendere questi dati valgono le stesse considerazioni fatte nelle due sezioni precedenti. Alla luce di ciò, risultano essere decisamente buoni, tenendo anche di conto che, utilizzando metodi di valutazione ed estrazione più sofisticati, può anche essere migliorato.

Per cui, questo nuovo metodo risulta essere decisamente valido per l'elaborazione di testi scientifici di dominio medico-sportivo, inoltre, può costituire uno strumento importante per ulteriori studi, come cercare di identificare le relazioni tra sport e malattie all'interno di testi.

Come per gli altri gruppi, mostro un'altra indagine che riporta ulteriori informazioni sulle estrazioni, in modo da dare un'idea più chiara del dataset.

	BioBERT	Manuale (malattie)	Baseline	Manuale (sport)
Entità estratte	244	243	126	128
Entità estratte correttamente	189	243	115	128
Entità per abstract	4,9	4,7	2,5	2,6
Entità corrette per abstract	3,8	4,7	2,3	2,6
Entità max per abstract	15	17	11	10
Entità max corrette per abstract	15	17	10	10

	BioBERT + Baseline	Manuale
Entità estratte	370	371
Entità estratte correttamente	304	371
Entità per abstract	7,4	7,4
Entità corrette per abstract	6,1	7,4
Entità max per abstract	16	19
Entità max corrette per abstract	15	19

Conclusioni

In conclusione, in questo lavoro sono state esplorate differenti metodi di estrazione delle entità nominate all'interno di testi scientifici di natura biomedica estratti ad hoc da PubMed.

Dopo aver calcolato precision, recall e f1-score di ciascun metodo sono stati scelti i due migliori: per l'estrazione delle malattie è risultato essere migliore il modello "jordyvl/biobert-base-cased-v1.2 ncbi disease-sm-first-ner" di bioBERT (precision = 0,78, recall = 0,82 e f1-score = 0,8), mentre per l'estrazione di sport, ha ottenuto risultati migliori il semplice pattern matching tramite la baseline composta di sport trovati sul web (precision = 0,877, recall = 0,945 e f1-score = 0,91). Infine, combinando le migliori tecniche è stato creato un metodo combinato di estrazione multi-dominio del tutto nuovo, in grado di estrarre sia malattie che sport, che, vedendone le statistiche, risulta essere decisamente buono con precision = 0,822, recall = 0,819 e f1-score = 0,821.

Questo nuovo metodo può essere la base di partenza per mettere appunto un metodo ancora più efficiente e, oltre ad essere un importante strumento di elaborazione di testi medico-sportivi in grado di estrarre informazioni come entità nominate, potrebbe essere utile per sviluppare un metodo di identificazione delle relazioni tra le entità estratte (relation extraction).

In generale un identificatore di relazioni permetterebbe di studiare relazioni multi-dominio e, in questo caso, si potrebbe indagare sull'esistenza di influenze negative dirette tra uno sport ed una malattia specifica, oppure se un particolare sport può essere utile alla cura di una malattia, oltre anche alla possibilità di effettuare varie indagini statistiche.

In aggiunta alla creazione di questo metodo, questo lavoro mette a disposizione una baseline composta da nomi di sport in lingua inglese ed un piccolo dataset annotato per entità medico-sportive, che possono essere utilizzati per sviluppare e testare nuovi metodi e modelli.

Bibliografia

- [1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Singler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- [2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [3] tloen Eric J. Wang. alpaca-lora. <https://github.com/tloen/alpaca-lora?tab=readme-ov-file>.
- [4] PubMed. Available from: <https://pubmed.ncbi.nlm.nih.gov/about/>.
- [5] PubMed Advanced Search Builder. Available from: <https://pubmed.ncbi.nlm.nih.gov/advanced/>.
- [6] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.
- [7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
- [8] jordyvl. jordyvl/biobert-base-cased-v1.2_ncbi_disease-sm-first-ner, 2022.
- [9] juletx Julen Etxaniz. alpaca-lora-mt. <https://github.com/juletx/alpaca-lora-mt>.
- [10] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. 02 2008.
- [11] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019.
- [12] Mona Lisa and Hew Bot. My Research Software, 12 2017.

- [13] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy, August 2019. Association for Computational Linguistics.
- [14] Shin SY. Current status and future direction of digital health in korea. *Korean J Physiol Pharmacol*, 2019.
- [15] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [16] Sayers E. A General Introduction to the E-utilities. 2009 May 26 [Updated 2022 Nov 17]. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK25497/>.
- [17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.
- [18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [19] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023.
- [20] Robert Wood. "list of sports - every sport from around the world." topend sports website, 2008.