UNIVERSITÀ DI PISA

Dipartimento di filologia
Corso di Laurea in Informatica umanistica

Deep Text Recognition

Relatori:
Prof. Claudio Gallicchio
Prof. Alessandro Lenci

Presentato da:
Carolina Sgherri

Sessione invernale
Anno accademico 2020/2021

Indice

Code Listings

Capitolo 1

Introduzione

Gli ultimi sviluppi tecnologici hanno consentito alle applicazioni digitali nei campi delle **Digital Humanities** e **Digital Knowledge** di avanzare in modi impensabili fino a qualche decina di anni fa, è di fondamentale importanza promuovere e sostenere l'uso delle ultime innovazioni in campo informatico come le tecnologie basate su Artificial Intelligence (AI) e Deep Learning (DL) per rendere il patrimonio culturale più facilmente accessibile e consultabile.

Il lavoro svolto in questa tesi mira, alla progettazione e allo sviluppo di un applicativo web, chiamata "Deep Text Recognition", realizzato tramite un insieme di varie tecnologie, illustrate maggiormente in dettaglio di seguito, in grado di effettuare il riconoscimento di caratteri manoscritti e l'elaborazione di questi ultimi al fine di rendere i dati fruibili in modo agevole e flessibile.
L'utilizzo di tale sistema potrebbe essere inoltre quello di facilitare un efficiente archiviazione di tali testi grazie alla possibilità di convertire i dati in formato XML/TEI, ciò potrebbe assistere l'utente nella creazione, o arricchimento di archivi e banche dati testuali.
L'applicativo permette inoltre l'esportazione in formato PDF (portable document format - formato di documento portabile) maggiormente indicato per le persone senza formazione tecnica specifica e TXT formato estremamente compatto, grazie alla rinuncia della formattazione del testo.

Gli obiettivi e le motivazioni della tesi sono quelli di testimoniare e dimostrare l'importanza della collaborazione fra i mondi dell'informatica e delle digital humanities per arrivare alla realizzazione di applicativi innovativi e in grado di esaltare il patrimonio culturale.

Ad oggi non sempre si riesce a raggiungere il massimo potenziale di questa collaborazione, ma con il tempo e il raffinamento degli strumenti si potrà arrivare a risultati sempre più raffinati.

All'interno capitolo numero 2, Artificial intelligence e Deep learning nelle Digital Humanities viene articolata una discussione sull'uso delle artificial inteligence e del deep learning nel campo delle Digital Humanities, con una breve disquisizione sulla digitalizzazione di documenti umanistici tramite queste tecnologie.
Ci sarà inoltre un focus sulle difficoltà e le problematiche nell'uso delle artificial intelligence nei campi di studio umanistici e di possibili soluzioni a queste difficoltà.

Nel capitolo numero 3, Background e Tecnologie usate vengono illustrate le tecnologie usate all'interno del progetto "Deep Text Recognition" suddivise in più sottocategorie:

- la prima sezione è dedicata al **machine learning**, una branca dell'artificial intelligence dedita allo studio di algoritmi in grado di migliorare automaticamente grazie all'esperienza, in particolare all'interno di questa sezione si va a illustrare l'uso e il funzionamento del deep learning, un campo del machine learning basato sull'uso di diversi livelli di rappresentazione e le artificial neural network.
- all'interno della seconda parte del capitolo viene invece illustrato il linguaggio di programmazione interpretato Python, che supporta multipli paradigmi di programmazione (procedurale, orientato agli oggetti, funzionale). All'interno della sezione dedicata a Python vengono inoltre descritte le varie librerie utilizzate all'interno del progetto
- la terza sezione dedicata alle tecnologie web si divide in tre macrosezioni dedicata ognuna a una fondamentale tecnologia per il funzionamento di un sito internet, l'HTML (HyperText Markup Language - linguaggio a marcatori per ipertesti), il linguaggio di markUp utilizzato per realizzare la struttura delle pagine web, il CSS (cascading style sheet - fogli di stile a cascata) il linguaggio usato per definire dei fogli di stile applicabili alle pagine HTML e infine JavaScript, linguaggio di programmazione di cui ci si serve per la creazione di effetti dinamici e interattivi all'interno delle pagine web
- nell'ultima sezione si descrive l'importanza della Text Encoding Initiative (TEI) e delle linee guida da loro create al fine di realizzare una codifica uniforme per i testi di interesse per le scienze umane e sociale, così da facilitarne l'elaborazione digitale

Per mezzo del capitolo numero 4, Applicazione,andremo a illustrare in mo-

do esauriente vari aspetti dell'applicazione oggetto di questa tesi "Deep Text Recognition", atta al riconoscimento di caratteri manoscritti all'interno di un'immagine di rilevanza umanistica al fine di poter usufruire di tale testo in vari formati.
Verranno inoltre esposte delle analisi dell'applicazione corredate di diagramma dei casi d'uso e di sequenza.
Infine, si potrà trovare una sezione riguardante l'implementazione di tale applicazione suddivisa in due sottosezioni: lato server, all'interno del quale viene illustrato l'utilizzo del linguaggio Python per la realizzazione dell'applicativo e lato client, dove si disquisisce dei vari framework e librerie utilizzate.

Nel capitolo numero 5, Guida all'uso dell'applicazione, verrà mostrato il corretto uso e funzionamento dell'applicativo realizzato, descritta nel capitolo precedente.
Per facilitare la comprensione delle istruzioni il capitolo segue la stessa suddivisione delle pagine del sito, quindi in quattro sottosezioni: home, la pagina principale del sito, da dove scegliere l'immagine da convertire, la seconda sezione è dedicata alla seconda pagina all'interno della quale è possibile fare la revisione del testo riconosciuto dall'OCR, all'interno dell'ultima pagina del flusso di modifica del testo è possibile inserire i metadati e usufruire del risultato dell'elaborazione in vari modi. Infine vi è la pagina About, raggiungibile tramite il menu in alto, all'interno della quale si possono trovare informazioni di carattere generale sul sito e la tesi.

All'interno del capitolo 6 potremo trovare le Conclusioni tratte e le note di sviluppi futuri dell'applicativo web.

Capitolo 2

Artificial intelligence e Deep learning nelle Digital Humanities

Le Digital Humanities sono sempre maggiormente orientate verso l'utilizzo delle AI[1] e del Deep learning per ridurre i tempi e i costi, allo stesso tempo esaltando e rendendo più accessibile il patrimonio culturale garantendo una conoscenza più approfondita delle varie tematiche culturali a un maggior numero di individui.
L'obiettivo della tesi è quello di dimostrare l'importanza della collaborazione fra i mondi dell'informatica e delle digital humanities per arrivare alla realizzazione di applicativi innovativi e in grado di esaltare il patrimonio culturale.

La definizione di Digital Humanities è continuamente riformulata da studiosi e professionisti, perché il campo è in continua crescita e cambiamento, quindi definizioni specifiche possono diventare rapidamente obsolete o limitarne inutilmente il potenziale futuro.
Le **Digital Humanities** (scienze umane digitali - informatica umanistica) sono quel campo di studi che integra le procedure computazionali e il sapere umanistico, non si tratta di un semplice utilizzo della tecnologia digitale come mero strumento, consiste bensì nella creazione di nuovi paradigmi e di rivoluzionarie strutture del pensiero critico.
In linea con il valore di rendere il patrimonio culturale maggiormente accessibile, molti progetti e riviste umanistiche digitali sono ad accesso aperto a

[1] Artificial Intelligence

dimostrazione dell'impegno del settore verso standard aperti e open source. [33]

Nonostante le radici della materia siano ormai risalenti la fine degli anni '40 del 900, quando padre Roberto Busa in collaborazione con IBM decise di indicizzare l'intera produzione letteraria di S. Tommaso d'Aquino [1], e di strada ne sia stata fatta tanta, di fatto oggi le Digital Humanities hanno abbracciato ogni campo del sapere umanistico (linguistica, arte, ...), ci sono ancora sfide ed opportunità sempre nuove ed entusiasmanti grazie all'evoluzione del digitale e all'ambito delle Digital Humanities che continua a espandersi. [33]

L'ente di coordinamento internazionale in cui si sono federate le più importanti associazioni di ricerca e promozione delle Digital Humanities è chiamata ADHO (Alliance of Digital Humanities Associations - alleanza delle associazioni di informatica umanistica). A partire dal 2006 l'ente organizza ogni anno una conferenza internazionale con funzioni di riferimento per il settore dell'informatica umanistica.
In Italia l'ente di riferimento nazionale è l'AIUCD (Associazione per l'Informatica Umanistica e la Cultura Digitale), che tiene congressi a partire dal 2012. [33]

Negli ultimi anni le **deep neural network** (DNN - reti neurali profonde) dominano il campo dell'analisi automatica del testo e quello dell'elaborazione naturale del linguaggio (natural language processing - NLP), presentando prestazioni anche sorprendenti. Le deep neural network sono sistemi di **deep learning** in grado di risolvere molte attività rilevanti per la ricerca nelle Digital Humanities, come il rilevamento della lingua di un testo, o il controllo ortografico.
Sebbene le tecniche di apprendimento classiche basate su regole, schemi, o caratteristiche predefinite non siano più considerate all'avanguardia in molte attività di elaborazione del testo, i ricercatori umanistici spesso ancora le usano.
Le **ragioni per evitare l'utilizzo del deep learning** potrebbero essere trovate nella mancanza di strumenti pronti all'uso su misura per l'attività necessaria, la mancanza di dati, tempo, formazione, risorse computazioni e limitazioni di budget. [22, 32]
Spesso però purtroppo nelle Digital Humanities un risultato viene considerato valido solo se è spiegabile il percorso necessario al suo raggiungimento tramite l'ermenautica classica delle scienze umane, tuttavia la maggior parte dei metodi computazionali non si presta ad essere spiegata in termini umanistici classici, quindi a causa di preconcetti e stereotipi, spesso questo genere di risorse vengono escluse da parte del personale umanistico a priori.[21]

Sono ancora moltissimi gli esperti dei campi umanistici che diffidano delle Digital Humanities sostenendo che il loro perseguire un'agenda rivoluzionaria, mini gli standard convenzionali di preminenza, autorità e potere disciplinare, compromettendo così i progressi possibili grazie a questo campo di studio. [33]

2.1 Digitalizzazione di documenti

Negli ultimi decenni, gli archivi di documenti storici cartacei sono stati digitalizzati utilizzando una tecnologia chiamata OCR (Optical Character Recognition - riconoscimento ottico dei caratteri).
L'Optical Character Recognition è un campo di ricerca nel pattern recognition (riconoscimento di schemi) dell'artificial intelligence.
Gli algoritmi di OCR sono software dedicati alla conversione del testo contenuto nelle immagini scansionate in testo machine readable (leggibile dal computer). Si tratta di un metodo comune di digitalizzazione, solitamente usato con testi stampati, in modo che possano essere modificati elettronicamente, ricercati, archiviati in modo più compatto e utilizzati in processi automatizzati. [37]

La qualità del testo convertito dagli OCR è una componente critica per la conservazione del patrimonio storico e culturale. Una qualità insoddisfacente significa che il testo non sarà ricercabile e analizzabile, o, peggio ancora, che l'analisi potrebbe portare a conclusioni errate.
Il riconoscimento di caratteri latini e dattiloscritti non è ancora accurato al 100% anche quando è disponibile un'immagine chiara. Il riconoscimento del testo corsivo è un'area di ricerca attiva con tassi di riconoscimento persino inferiori di quelli precedentemente riportati.
Gli OCR generici ottengono risultati ancora meno accurati se applicati a testi storici. [22, 37]

La post-correzione di archivi storici digitalizzati su piccola scala o in lingue di nicchia è una sfida che può essere risolta solamente utilizzando modelli di deep learning con elevata precisione, tecnica applicabile solamente nel caso in cui sia possibile ottenere un set di dati appropriato. [22]

2.2 Problematiche nell'uso del deep learning nelle Digital Humanities

I modelli di deep learning sono solitamente sviluppati da scienziati informatici e sono addestrati, testati e ottimizzati per scopi generici, o commerciali da parte di imprese che si occupano di testi moderni, nonostante le risorse delle

Digital Humanities siano anch'esse testuali, l'adattamento di tali modelli non è così semplice.
Le principali sfide nell'uso del deep learning per l'analisi di risorse di tipo umanistico sono due:

1. la **mancanza di dati** con cui addestrare le reti neurali, anche quando c'è un grande corpus di testo, non ci sono set di dati etichettati in modo bilanciato, cioè con la giusta distribuzione di esempi giusti e sbagliati e sono necessari cambiamenti o adattamenti nell'architettura della rete per ottenere una buona precisione per tali set di dati.
   Questa problematica colpisce anche gli algoritmi di apprendimento automatico classici del machine learning, anche se in modo più marginale, visto che gli algoritmi di deep learning supervisionati richiedono una quantità significativamente maggiore di dati rispetto a quelli di machine learning, questo è uno dei motivi per cui quando i ricercatori utilizzano il deep learning spesso utilizzano algoritmi non supervisionati, che non richiedono dati di addestramento e sono limitati a compiti specifici
2. **necessità di adattare il dominio d'uso**, in molte attività considerate comuni nell'elaborazione del linguaggio naturale, l'interpretazione delle Digital Humanities di tale attività e di conseguenza la sua realizzazione è diversa. Inoltre, le risorse delle Digital Humanities potrebbero dover essere pre-elaborate prima di poter essere usate come input per i modelli di deep learning [22, 32, 36]

Per queste motivazioni c'è la necessità non solo di trovare un ponte di comunicazione fra i conoscitori delle Digital Humanities e gli esperti informatici in grado di personalizzare e sviluppare tecnologie, ma c'è inoltre l'esigenza di formare nuovi specialisti appartenenti alla comunità delle Digital Humanities in grado di dare vita a nuova tecnologia e rinnovare quella già esistente per meglio adattarla al loro ambito di studio.

2.2.1 Soluzioni

Ci sono varie soluzioni adottabili per cercare di aggirare la mancanza di dati, illustrate di seguito le principali.

Un corpus corretto può essere prodotto tramite una **procedura di crowdsourcing** [2] tali informazioni possono servire come set di dati di addestramento per gli algoritmi di deep learning.

[2]Richiesta di idee, suggerimenti, opinioni, rivolta agli utenti di Internet.

Tuttavia, sebbene il metodo del crowdsourcing dia risultati soddisfacenti, è purtroppo adatto principalmente per lingue abbondantemente diffuse come l'inglese, inoltre generare manualmente un set di dati per addestrare un modello di deep learning è costoso e inefficiente, anche con l'attività di crowdsourcing. [22]

Una metodologia più efficiente per risolvere il problema della mancanza di dati prevede un **approccio in due fasi**, nella prima fase gli esseri umani vengono utilizzati per creare una piccola serie di esempi, questa piccola serie viene poi utilizzata nella seconda fase da un insieme di algoritmi per generare un dataset contenente numerosi esempi di training.
Questa metodologia può essere applicata in due diversi modi:

- cercando **pattern ricorrenti** in un piccolo numero di esempi corretti manualmente e usarli in seguito per generare automaticamente più esempi corretti
- con l'approccio della **distant supervision** (supervisione a distanza), un classificatore viene addestrato su un piccolo insieme di esempi e applicato a un grande corpus. Il classificatore classificherà i dati con un'accuratezza relativamente bassa, ma sufficientemente elevata da consentire al modello di apprendere altre caratteristiche da questa classificazione debole.
  Questo metodo è anche stato utilizzato nella sentiment analysis (analisi dei sentimenti) sui dati delle testimonianze dell'olocausto. [22]

Un approccio diverso per risolvere il problema della mancanza di dati è il metodo del **transfer learning** (apprendimento per trasferimento).
Nel transfer learning si crea un modello di addestramento su un dataset generico, adatto all'attività da risolvere, ma con dati di un altro dominio, o generici. Il modello generico viene quindi nuovamente addestrato utilizzando un piccolo insieme di esempi specifici.
Questo metodo può inoltre risolvere la seconda problematica incontrata precedentemente, quella della necessità di adattare il dominio d'uso, grazie all'utilizzo iniziale di un dataset generico con cui allenare il modello. [22]

Capitolo 3

Background e Tecnologie usate

All'interno di questo capitolo verranno illustrate le tecnologie utilizzate all'interno del progetto "Deep Text Recognition"; il capitolo si suddivide in sottosezioni dove le tecnologie sono divise per tipologia e linguaggio di programmazione.

La sezione 3.1 affronta il **machine learning**, una branca dell'artificial intelligence che studia algoritmi in grado di migliorare automaticamente attraverso l'esperienza e l'uso dei dati e del **deep learning** un campo del machine learning, che si basa su artificial neural network. [3]
La sezione 3.2 tratta invece il linguaggio di programmazione general-purpose **Python**, particolarmente usato per il machine learning e il deep learning grazie alla grande disponibilità di supporto dato dalla grande comunity di utenti e dalle numerose librerie esistenti. [10]
La sezione 3.3 di questo capitolo analizza le varie **tecnologie web** utilizzate all'interno del progetto, tale sezione si divide in tre grandi sottocategorie:

- **HTML** (HyperText Markup Language - linguaggio a marcatori per ipertesti), il linguaggio di Markup utilizzato per realizzare le pagine web attraverso dei tag
- **CSS** (cascading style sheet - fogli di stile a cascata), il linguaggio utilizzato per definire la formattazione delle pagine web
- **JavaScript**, linguaggio di programmazione solitamente usato all'interno delle pagine web per la realizzazione di effetti dinamici e interattivi [17, 16, 18]

All'interno della sezione 3.4 di questo capitolo viene illustrata invece l'importanza della Text Encoding Initiative (**TEI**) e vengono descritte nel dettaglio le **linee guida**, da loro create, riguardanti la codifica di testi di interesse per le scienze umane e sociali, al fine di facilitarne l'elaborazione automatica, indipendentemente dalla piattaforma utilizzata. [5, 41]

3.1 Machine Learning

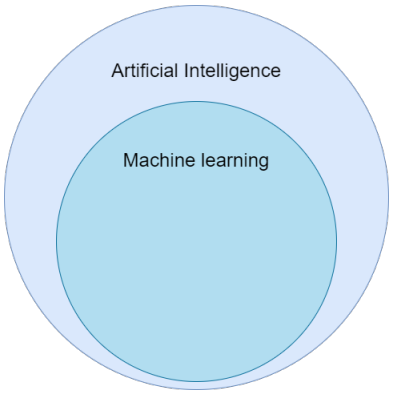


Figura 3.1: Come il machine learning è una sottocategoria dell'artificial intelligence (AI)

Prima di poter parlare di machine learning va introdotto il concetto di artificial intelligence (Figura 3.1). L'**artificial intelligence** (intelligenza artificiale) è lo studio della progettazione di intelligent agents (agenti intelligenti). Un intelligent agent è un sistema che agisce in modo intelligente, che intraprende cioè azioni appropriate al contesto, per il raggiungimento dei propri obiettivi, impara dall'esperienza e fa scelte appropriate date le limitazioni percettive e computazionali. [24]

Il **machine learning**, detto anche apprendimento automatico, è lo studio di algoritmi in grado di migliorare automaticamente, grazie all'esperienza e l'uso dei dati. [36]
Si tratta di un campo di ricerca all'intersezione tra statistica, artificial intelligence e informatica ed è infatti anche conosciuto come **analisi predittiva**, o **apprendimento statistico**. [19]

Gli algoritmi di machine learning sono utilizzati in un'ampia varietà di **applicazioni onnipresenti nella vita di tutti i giorni**, nella medicina, grazie all'apprendimento automatico possiamo oggi determinare se un tumore è benigno e analizzare sequenze di DNA, grazie allo speech recognition (riconoscimento vocale) abbiamo la sottotitolazione automatica e sempre più assistenti vocali (Google, Alexa, Cortana, ...) e è per esempio merito della computer vision (visione artificiale) se la maggior parte dei social network, come Facebook, sono in grado di riconoscere i volti nelle foto. [19]

Il **data mining** è l'applicazione di tecniche di Machine Learning per esplorare grandi quantità di dati per estrarne informazioni strutturate. [11]

3.1.1 Storia

Agli albori delle applicazioni "intelligenti", la maggior parte dei sistemi usava, istruzioni "if" e "else" codificate a mano per elaborare i dati, o adeguarsi all'input utente, ma **creare manualmente regole decisionali** non è fattibile per determinate applicazioni, perché presenta due grandi svantaggi:

- la logica richiesta per prendere una decisione è specifica per un singolo dominio e attività. La modifica anche minima di tale attività potrebbe richiedere una riscrittura dell'intero sistema
- la progettazione di regole richiede una profonda comprensione di come una decisione dovrebbe essere presa da un essere umano esperto. [19]

Sebbene il machine learning abbia iniziato a prosperare solo negli anni '90, è rapidamente cresciuto, al punto di diventare il sottocapo più popolare e di maggior successo dell'artificial intelligence, una tendenza guidata dalla disponibilità di hardware più veloce e di dataset più grandi. [3]

3.1.2 Funzionamento

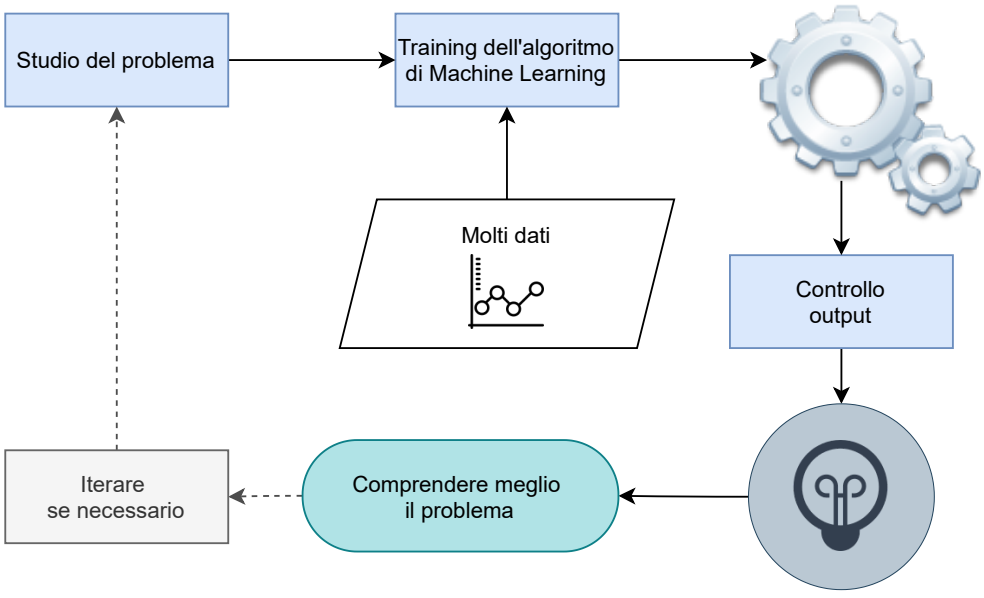


Figura 3.2: Funzionamento del machine learning.

La disciplina dell'apprendimento automatico utilizza vari approcci per insegnare ai computer a svolgere attività in cui non è disponibile un algoritmo

tradizionale, o non è completamente soddisfacente. Esistono quattro categorie principali di algoritmi, tutti con in comune i passaggi in figura 3.2. Tali categorie sono:

- **apprendimento supervisionato** –> i dati di addestramento forniti all'algoritmo includono l'output desiderato
- **apprendimento non supervisionato** –> i dati di addestramento non sono etichettati, il sistema cerca di etichettare i dati raggruppandoli per qualche misura di similarità
- **apprendimento semi-supervisionato** –> in quanto l'etichettatura dei dati richiede solitamente tempo e denaro, spesso non tutte le istanze sono etichettate, gli algoritmi di questa categoria possono gestire dati parzialmente etichettati
- **apprendimento per rinforzo** –> il sistema di apprendimento, chiamato agente in questo contesto, può osservare l'ambiente, selezionare, eseguire azioni e ottenere ricompense in cambio. Deve poi imparare da solo quale è la migliore strategia, detta policy, per ottenere la ricompensa più grande [11, 3]

Ogni algoritmo è differente in termini di quali tipi di dati e tipologia di problemi risolve meglio.
Per questo una delle parti più importanti del processo di machine learning è comprendere i dati con cui stai lavorando e come questi si relazionano con l'attività da svolgere.
Costruire una buona rappresentazione di dati è detto **feature selection** (selezione delle funzionalità), o **feature engineering** (ingegneria delle funzionalità). [19]

Gli algoritmi di machine learning più di successo sono quelli che automatizzano il processo decisionale **generalizzando da esempi conosciuti**, si parla cioè di **apprendimento supervisionato**.
Ciò si basa sul fatto che è probabile che strategie, algoritmi e inferenze, che hanno funzionato bene in passato continueranno a funzionare bene in futuro.
In questa metodologia, un utente fornisce all'algoritmo insieme di coppie di input e relativo output, detto **training set**, e l'algoritmo trova in questi esempi una struttura statistica al fine di consentire al sistema di elaborare delle regole per automatizzare l'attività. [36, 19, 3]

Anche se creare **dataset di input e output** è spesso un processo manuale laborioso, gli algoritmi di apprendimento supervisionato sono ben compresi e le loro performance sono di facile misurazione, per questo vengono spesso preferiti. [36, 19]

L'unico modo per valutare le performance di un algoritmo è la valutazione sul **test set**.
Se un modello è in grado di fare previsioni accurate su dati mai visti, si dice in grado di **generalizzare dal training set al validation set**.
Se un modello è eccessivamente complesso rispetto al quantitativo di informazioni possedute è detto **overfitting**. L'overfitting avviene quando si adatta troppo un modello alle particolarità del training set e si ottiene così un modello che funziona perfettamente sul training set, ma non è in grado di generalizzare su nuovi dati.
D'altro canto, se il modello è invece troppo semplice allora si potrebbe non essere in grado di catturare tutti gli aspetti e la variabilità dei dati e il modello darà pessimi risultati anche sul training set. Scegliere un modello troppo semplice è detto **underfitting**. [19]

3.1.3 Artificial neural network

Le **artificial neural network** (reti neurali), o dette anche semplicemente neural network (Figura 3.3), sono dei modelli computazionali ispirati alla semplificazione di una rete neurale biologica utilizzati per cercare di risolvere problemi come quelli di artificial intelligence.
Sono nate come un tentativo di sfruttare l'architettura del cervello umano per eseguire compiti con cui gli algoritmi convenzionali hanno avuto scarso successo, ben presto si decise di abbandonare il tentativo di rimanere fedeli ai precursori biologici preferendo l'uso di reti a forma di grafo orientato e ponderato. [30]

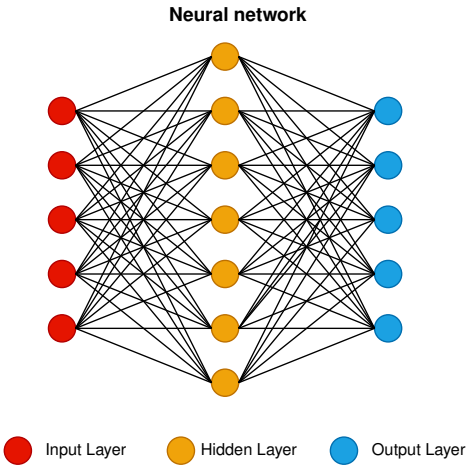


Figura 3.3: Esempio di rete neurale.

Le artificial neural network si basano su un insieme di nodi collegati fra di loro chiamati neuroni, che astraggono idealmente i neuroni in un cervello biologico.
I neuroni possono ricevere segnali e generare come output, verso neuroni del successivo livello, il risultato di una funzione calcolata sulla base dei segnali ricevuti, la funzione utilizzata per calcolare tale output è detta **funzione di attivazione**.
I collegamenti fra neuroni vengono detti **archi**, ogni arco ha un peso numerico, che si adatta man mano che l'apprendimento procede; il peso aumenta, o diminuisce la potenza del segnale. [30]

Solitamente i neuroni sono aggregati in **layer** (strati), strati diversi possono eseguire trasformazioni diverse sui loro input.
I segnali viaggiano dal primo layer, quello di input, all'ultimo layer, quello di output, possibilmente dopo aver attraversato i layer più volte. [30]

3.1.4 Deep Learning

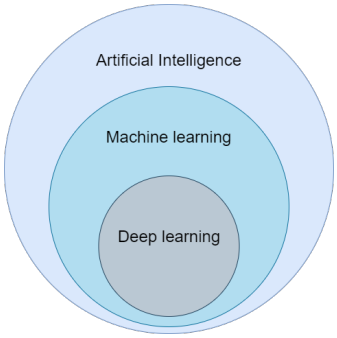


Figura 3.4: Deep learning sottoinsieme del machine learning a sua volta sottoinsieme dell'AI.

Il **Deep Learning** è un sottocampo del machine learning (Figura 3.4), basato sulle artificial neural network, che pone l'accento sull'apprendimento in livelli successivi di rappresentazioni significative.
L'aggettivo deep (profondo) in deep learning, si riferisce a questa idea di strati successivi di rappresentazione. [3, 19, 32]

Il numero di layer che contribuisce a un modello è chiamato **profondità**. Il deep learning moderno spesso coinvolge decine, o addirittura centinaia di livelli successivi di rappresentazioni e vengono tutti appresi automaticamente dall'esposizione ai dati di addestramento (training set). [3]

Nel deep learning queste rappresentazioni stratificate vengono, quasi sempre, apprese tramite modelli chiamati reti neurali, strutturate per livelli di neurone, come in figura 3.3 a pagina 26. Nonostante la confusione che viene però spesso fatta dai media non specialistici e nonostante i concetti tradizionali di deep learning siano stati sviluppati in parte traendo ispirazione dalla nostra comprensione del cervello, i modelli di deep learning non sono modelli di cervello. [3]

Uno dei principali **vantaggi** degli algoritmi di deep learning è che sono in grado di estrarre informazioni in enormi quantità di dati e di costruire modelli estremamente complessi. Dato abbastanza tempo di computazione, dati e un'attenta messa a punto (tuning) dei parametri, spesso superano in prestazioni gli altri algoritmi di machine learning, specialmente nei compiti di:

- **classificazione** –> predire un'etichetta scelta fra una lista predefinita di possibilità, come per esempio classificare le e-mail come "spam" o "non spam"
- **regressione** –> prevedere un numero continuo di numeri in virgola mobile, come per esempio predire il reddito annuo di una persona a partire dalla loro educazione, età, indirizzo di residenza.

Di controparte fra gli **svantaggi** di questi algoritmi, troviamo il fatto che i tempi

di addestramento (training) sono molto lunghi e inoltre richiedono un'attenta preelaborazione dei dati. [19]

3.1.4.1 Funzionamento

Come già precedentemente detto un modello di machine learning riguarda la mappatura degli input sui relativi output attraverso una sequenza di trasformazione matematiche successive (layer) i cui parametri sono estratti dal modello durante la fase di addestramento. [19, 3, 32]

La specifica di ciò che un livello fa ai suoi dati di input è memorizzata nei pesi di quel livello, talvolta chiamati anche **parametri** di un livello.
In questo contesto imparare significa trovare un insieme di valori per i pesi di tutti i livelli in una rete, in modo che la rete mappi correttamente gli input di esempio agli output associati. [3]

Una deep neural network può però contenere decine di milioni di parametri, trovare il valore corretto per tutti può sembrare impossibile, soprattutto perché la modifica di anche solo il singolo peso di un arco si ripercuote sull'intera rete neurale.
Ogni volta che durante la fase di allenamento, dopo alcuni tentativi di predizione se ci sono degli errori è necessario aggiornare i pesi dei singoli archi di ogni nodo per fare in modo che la rete neurale faccia la predizione corretta. Per stimare come e di quanto i pesi degli archi debbano essere modificati si usa una **loss function** (funzione di perdita), che stima la distanza tra la predizione fatta dalla rete neurale e l'output corretto. [3]

Il trucco fondamentale nel deep learning è usare questo punteggio come **segnale di feedback** per regolare leggermente il valore dei pesi, in una direzione che abbasserà il punteggio di perdita. Questa regolazione è compiuta dell'ottimizzatore, che implementa quello che viene chiamato l'**algoritmo di Backpropagation** (retropropagazione), l'algoritmo centrale del deep learning. [3]

Inizialmente ai **pesi** della rete vengono assegnati valori **casuali**, quindi la rete implementa solamente una serie di trasformazioni casuali, naturalmente l'output è lontano da quello desiderato e il punteggio della funzione di perdita è di conseguenza molto alto, ma con ogni esempio, che la rete elabora, i pesi vengono leggermente modificati e il punteggio della funzione di perdita diminuisce.
Questa inizializzazione casuale influenza il modello che viene appreso, ciò significa, che anche usando esattamente gli stessi parametri, si possono ottenere

risultati diversi. Se le reti sono molto grandi e la loro complessità è scelta accuratamente, questo non dovrebbe avere effetti troppo grandi sull'accuratezza, ma è importante tenerne di conto, specialmente nel caso di piccole reti. [19, 3]

Questo è il training loop (ciclo di addestramento) che, ripetuto un numero sufficiente di volte (in genere decine di iterazioni su migliaia di esempi), imposta valori di peso che riducono al minimo il punteggio della funzione di perdita. [19, 3]

3.1.4.2 Principali architetture

Una **deep neural network** (rete neurale profonda) è una rete neurale artificiale con molti livelli, oltre quelli di input e di output.
In figura 3.5 possiamo osservare le differenze nel numero di livelli fra le neural network e le deep neural network.

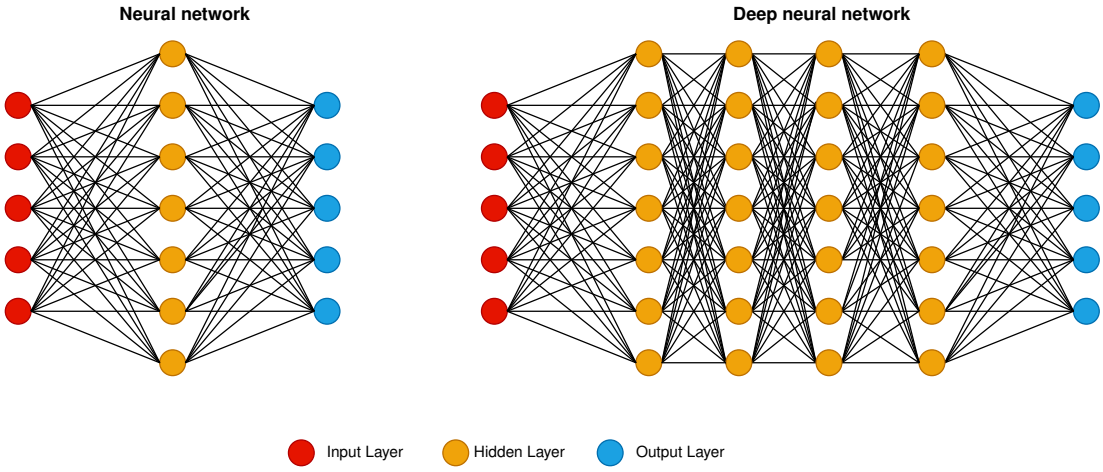


Figura 3.5: La differenza tra una rete neurale semplice e una profonda

Come ogni altra rete neurale è costituita da neuroni e sinapsi, che possono essere addestrati come qualsiasi altro algoritmo di Machine Learning.

Queste tipologie di reti sono generalmente **feedforward**, i dati fluiscono cioè dal livello di input al livello di output senza effettuare loopback. [32]

Le **recurrent neural network** (RNN - reti neurali ricorrenti) sono artificial neural network in cui i dati possono fluire in qualsiasi direzione, vengono usate per applicazioni come la modellazione del linguaggio. [32]

Le **long short term memory** (LSTM - memoria a lungo breve termine) sono artificial neural network dove a differenza delle reti neurali feedforward, i dati possono anche fluire in direzione inversa. Per questo motivo sono consigliate per attività come speech recognition e handwriting recognition. [32]

3.1.4.3 Storia

Nel 2006 Geoffrey Hinton pubblicò un articolo dove mostrava come addestrare una deep neural network in grado di riconoscere le cifre scritte a mano con un'elevatissima precisione (>98%). Nominarono questa tecnica come Deep Learning.
L'addestramento di una deep neural network era all'epoca considerato impossibile, perché troppo dispendioso computazionalmente, la maggior parte dei ricercatori aveva abbandonato l'idea verso la fine degli anni '90.
Questo articolo ha riacceso l'interesse della comunità scientifica e in poco tempo molti nuovi articoli hanno dimostrato che non solo il Deep Learning era possibile, ma anche capace di risultati incredibili, che nessun'altra tecnica di Machine Learning avrebbe potuto ottenere. [11]

Questa è un'area del machine learning in **rapida evoluzione**, con innovazioni e nuove applicazioni annunciate settimanalmente. Le recenti scoperte nell'apprendimento automatico e nell'artificial intelligence, come la vittoria del software Alpha Go contro i campioni umani nel gioco del Go, il miglioramento costante delle prestazioni della comprensione del parlato e la disponibilità di una traduzione vocale quasi istantanea, sono state tutte guidate da questi progressi. [19]

3.2 Python

Python è un linguaggio di programmazione **interpretato** e **general purpose**, la cui filosofia di design enfatizza la leggibilità del codice grazie all'uso dell'**indentazione**, al posto delle parentesi, o delle parole chiave, per delimitare i blocchi, in questo modo la struttura visiva del programma rappresenta accuratamente la struttura semantica del programma. [39, 10, 19]
Di seguito del codice di esempio che stampa in console la stringa "Hello world!" (Code Listing 3.1).

```
1 def funzioneHelloWorld():
2     print("Hello world!")
```

```
3
4 funzioneHelloWorld()
```

Code Listing 3.1: Esempio di Python

Questo linguaggio supporta **molteplici paradigmi** di programmazione come quello strutturata, orientato agli oggetti e funzionale.
Piuttosto che avere tutte le sue funzionalità integrate nel nucleo, Python è stato progettato per essere altamente estensibile con dei moduli. Grazie alla sua libreria standard completa è spesso descritto come un linguaggio "**batteries included**". [39, 10]

La filosofia di base del Python è riassunta nel documento "**The Zen of Python**", che include aforismi come:

- Beautiful is better than ugly - bello è meglio che brutto
- Explicit is better than implicit - esplicito è meglio che implicito
- Simple is better than complex - semplice è meglio che complesso
- Complex is better than complicated - complesso è meglio di complicato
- Readability counts - la leggibilità conta [23]

Fu ideato da **Guido Van Rossum** alla fine degli anni 80' presso il Centrum Wiskunde & Informatica (CWI) nei Paesi Bassi come successore del linguaggio di programmazione ABC.
Van Rossum ha avuto la responsabilità esclusiva del progetto, fino al 2018, quando ha annunciato la sua vacanza permanente dalle sue responsabilità di benevolo dittatore a vita del Python, un titolo conferitogli dalla Python comunity, nel 2019 è stato eletto un consiglio direttivo di cinque membri per guidare il progetto. [39]

Il linguaggio di programmazione Python è diventata la lingua franca della maggior parte delle applicazioni di data science, perché combina il potere dei linguaggi di programmazione general-purpose, con la facilità d'uso di linguaggi di scripting di specific-domain come R, o Matlab.
In quanto a linguaggio di programmazione general-purpose Python permette inoltre di creare facilmente user interfaces (GUIs) complesse e web services, inoltre permette una più semplice integrazione con sistemi preesistenti. [19]

Python ha librerie per il caricamento e la visualizzazione di dati, statistica, l'elaborazione del linguaggio naturale, elaborazione di immagini e molto altro. Questa vasta varietà di strumenti fornisce ai data scientist una grande quantità di funzionalità.

Uno dei principali vantaggi di usare Python è la possibilità di interagire direttamente con il codice, usando il terminale o altri strumenti come i notebook. [19, 39, 10]

3.2.1 Flask

Flask è un **microframework Python**, rilasciato sotto la licenza BSD5. Nonostante sia un microframework e quindi non supporti componenti in cui librerie di terze parti forniscono funzioni comuni, supporta delle **estensioni** che possono aggiungere funzionalità come se fossero implementate dallo stesso Flask. Un gran numero di applicazioni si basa su Flask come Pinterest e LinkedIn. Flask è inoltre compatibile con numerosi servizi Platform-as-a-Service, come Google App Engine o Heroku.
A partire da Ottobre 2020 Flask ha ricevuto il maggior numero di stelle su GitHub tra i framework di sviluppo web per Python ed è stato votato il framework più popolare nel Python Developers Survey del 2018. [7, 34]
Di seguito uno snippet di codice Python, che utilizza Flask per la realizzazione di un applicativo che espone sulla route "/home" una pagina contenente la stringa "Hello World!" (code listing 3.2).

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route("/home")
5 def hello():
6     return "Hello World!"
7
8 if __name__ == "__main__":
9     app.run()
```

Code Listing 3.2: Esempio di uso di Flask

3.2.1.1 Storia

La prima release di Flask risale al 1° Aprile 2010, ad opera dello sviluppatore austriaco membro del gruppo Pocoo, un gruppo di appassionati di Python formatosi nel 2004.
Inizialmente l'idea era quella di un pesce d'Aprile, ma l'idea divenne popolare

al punto di trasformarsi in una reale applicazione. Anche il nome Flask è un gioco di parole basato sul nome del framework, a cui si ispira, Bottle.
Nell'Aprile del 2016 il gruppo Pocoo è stato smantellato e lo sviluppo di Flask e delle relative librerie è passato all'appena formato Pallets Project. [34]

3.2.1.2 Componenti

Flask fa parte dei Pallets Projects (precedentemente Pocoo) e si basa su molti altri di essi:

- **Werkzeug** –> un toolkit per applicazioni Web Server Gateway Interface (WSGI) e come Flask è stata pubblicata sotto licenza BSD. Werkzeug può realizzare oggetti per richieste, risposte e utility functions
- **Jinja** un template engine sviluppato da Ronacher e rilasciato sotto licenza BSD, che gestisce i template in un sandbox
- **MarkupSafe** –>una libreria di gestione delle stringhe, rilasciato sotto licenza BSD. MarkupSafe estende il tipo stringa Python e si assicura il suo contenuto sia "sicuro"
- **ItsDangerous** –> una libreria di serializzazione dati sicura, rilasciato in licenza BSD. Viene utilizzato per memorizzare la sessione di un'applicazione Flask in un cookie senza consentire agli utenti di manomettere i contenuti delle sessioni [34]

3.2.2 Modulo Xml

Xml è un modulo Python utilizzato per la rappresentazione di dati nell'omonimo formato.
Questa libreria standard non può convalidare la struttura del documento rispetto a uno schema (XS, DTD), per queste funzionalità vanno usate librerie terze.

I moduli di elaborazione xml non sono protetti contro i dati costruiti in modo dannoso, un utente malintenzionato può abusare delle funzionalità xml per aggirare il codice e eseguire attacchi DoS (denial of service), accedere ai file locali, generare connessioni di rete ad altre macchine, o aggirare firewall. [10]

3.2.3 PIL

PIL (Python Imaging Library - libreria Python per le immagini) è una libreria open source per il linguaggio di programmazione Python, che aggiunge il supporto per l'**apertura**, la **manipolazione** e il **salvataggio** di file immagine in vari formati (PPM, PNG, JPEG, GIF, TIFF, ...). [4, 38]

Lo sviluppo del progetto originale PIL è stato interrotto è stato interrotto nel 2012, successivamente sono passati allo sviluppo del progetto Pillow, derivato da PIL, con supporto alle versioni 3.x del Python. Pillow è stato adottato al posto di PIL anche nelle distribuzioni Linux, in particolare Debian e Ubuntu. [10, 38]

Pillow offre diverse procedure standard per la manipolazione di immagini, per esempio: [38, 4]

- manipolazione per-pixel
- gestione di maschere e trasparenze
- filtri come sfocatura, rilevamento dei bordi, ...
- miglioramento di immagini, nitidezza, regolazione di luminosità, contrasto, ...
- conversione di immagini da un formato al altro

Di seguito una sezione di codice Python in cui grazie a PIL viene aperta l'immagine "file.jpg" e le viene applicato un filtro blur (Code Listing: 3.3).

```
1 from PIL import Image, ImageFilter
2 image = Image.open("file.jpg")
3 blur_image = original_image.filter(ImageFilter.BLUR)
```

Code Listing 3.3: Esempio di uso di PIL

3.2.4 PyFpdf

PyFPDF è una libreria per la generazione di documenti PDF in Python, si tratta di un porting della classe FPDF per PHP. [9, 8]
La libreria non ha nessuna dipendenza esterna, o estensione, ma offre la possibilità di utilizzare PIL per il supporto all'utilizzo delle GIF.

Di seguito una sezione di codice Python che utilizza PyFpdf per creare un pdf di una pagina contenente la stringa di testo "Hello world!" allineata a sinistra, tale pdf viene poi salvato nel file "file.pdf" (Code Listing: 3.4).

```
1 pdf = FPDF()
2 pdf.add_page()
3 pdf.cell(200, 10, "Hello world!", ln=1, align='L')
4 pdf.output("file.pdf")
```

Code Listing 3.4: Esempio di uso di PyFpdf

3.2.5 PyJwt e Uuid

PyJwt è una libreria che permette di **codificare e decodificare JSON Web Tokens**(JWT). Il **JWT** è un standard internet aperto (RFC 7519) utilizzato per la rappresentazione di dati con firma e/o crittografia, entrambe opzionali, il cui payload contiene del JSON, che asserisce un certo numero di affermazioni. [29]

Di seguito uno snippet di codice Python all'interno del quale viene utilizzato PyJwt per codificare e decodificare con l'algoritmo H256 e la chiave "secret" il Json ""payload": "Hello world!"" (Code Listing: 3.5).

```
1 import jwt
2 encoded = jwt.encode({"payload": "Hello world!"},"secret",
     algorithm="HS256")
3 jwt.decode(encoded, "secret", algorithms=["HS256"])
```

Code Listing 3.5: Esempio di uso di PyJwt

Questa libreria può essere usata in concomitanza con il modulo python UUID, per il riconoscimento univoco di un utente che visita un sito internet senza violare la loro privacy, tenendo conto per esempio il loro indirizzo IP e senza obbligare gli utenti a registrarsi al sito.

Il modulo Python UUID fornisce oggetti UUID immutabili e le funzioni uuid1(), uuid3(), uuid4(), uuid5() servono per generare rispettivamente UUID versione 1, 3, 4 e 5 come specificato nello standard RFC 4122.
La funzione uuid1() può compromettere la privacy, poiché crea un UUID contenente l'indirizzo di rete del computer; quindi, se si desidera creare un ID

univoco è più sicuro utilizzare la funzione uuid4(), che crea un UUID casuale. [10]

Di seguito uno snippet di codice Python all'interno del quale viene utilizzato il modulo Uuid per creare un Uuid casuale (Code Listing: 3.6).

```
1 uuid.uuid4()
```

Code Listing 3.6: Esempio di uso del modulo Uuid

3.2.6 Tesseract OCR

Tesseract è un text recognition Engine (OCR - Optical character recognition); si tratta di un progetto open source, avviato da Hewlett-Packard (HP) e successivamente preso in carico da Google.
Dalla versione 4.0 è stato aggiunto un nuovo OCR engine basato su reti neurali LSTM (precedentemente citate nella sezione 3.1.4.2). [28]
La versione originale di Tesseract consente di utilizzare le API in C e C++, ma esistono dei progetti di terze parti (come training projects e wrappers), che consentono di usare l'engine anche con le API di altri linguaggi; all'interno della tesi è stato usato un wrapper Python della libreria chiamato TesserOCR. TesserOCR si integra direttamente con le API C++ di Tesseract utilizzando il linguaggio Cython per consentire un codice semplice e performante. [10][6]

3.2.6.1 Origini

Tesseract fu inizialmente sviluppato tra il 1985 e il 1994 come software proprietario della Hewlett-Packard (HP), per i loro scanner, ma non è mai stato utilizzato dai loro prodotti.
In seguito nel 1996 è stato eseguito il porting su Windows e nel 1998 sono state fatte alcune migrazioni da C a C++.
In un test dell'Università del Nevada, Las Vegas (UNLV), Tesseract è emerso come uno degli OCR più precisi nel 1995, per questo quando nel 2005, dopo che HP ha lasciato il mercato degli OCR, lo sviluppo è stato consegnato all'Information Science Research Institute dell'UNLV.
Lo sviluppatore di HP, Ray Smith, nell'originale team di sviluppo di Tesseract, nel 2005 ha ricominciato a lavorare a Tesseract, alla Google, lo ha aggiornato e lo ha rilasciato nello stesso anno con la licenza Apache tramite Source Forge. Il codice è poi migrato da SourceForge alla piattaforma di sviluppo software

di Google, chiamata Google Code, dal 2015 sono invece passati a GitHub. Dal 2006 Tesseract viene usato come base per Google Books. Oggigiorno inoltre Google, utilizza Tesseract per il riconoscimento del testo sui dispositivi mobili e nei video, nonché per il rilevamento dello spam nelle immagini delle e-mail. Alla fine del 2016, Tesseract, versione 4, ha introdotto una rete neurale per il riconoscimento del testo. Alla fine del 2020 la biblioteca digitale non profit Internet Archive ha cambiato il riconoscimento automatico del testo da ABBYY FineReader a Tesseract, che elabora quindi più di 2 milioni di pagine al giorno. [28, 25]

3.2.6.2 Perché Tesseract

All'inizio della progettazione dell'applicativo web "Deep Text Recognition", sono stati testati diversi OCR in modo da trovare il più efficace, gli OCR (Optical character recognition - riconoscimento ottico dei caratteri) testati erano:

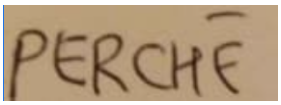
- **Keras OCR** –> modello OCR creato con architettura CRNN (convolutional recurrent neural network - rete neurale ricorrente convoluzionale), che utilizza TensorFlow rilasciata sotto licenza Apache 2.0 all'interno del TensorFlow Hub [14]
- **Rosetta** –> modello OCR completamente convoluzionale, che utilizza Torch e Tensorflow e un'architettura ResNet (Residual Network - rete residua), le architetture ResNet hanno notevolmente migliorato le prestazioni delle artificial neural network grazie all'aggiunta di più livelli. Questo modello è rilasciato sotto licenza Apache 2.0 all'interno del TensorFlow Hub [26]
- **Tesseract** –> illustrato precedentemente nel capitolo 3.2.6

È stato inoltre testato anche un HTR (Handwritten Text Recognition - riconoscimento del testo scritto a mano):

- **Simple HTR** –> questo HTR implementato con TensorFlow consiste di 5 livelli CNN (convolutional neural network - rete neurale convoluzionale), 2 livelli RNN (recurrent neural network - rete neurale ricorrente) LSTM (long short term memory - memoria a lungo termine) e infine CTC loss (Connectionist Temporal Classification Loss) e un decoding layer (livello di decodifica). Questo modello è rilasciato sotto licenza MIT su GitHub [12]

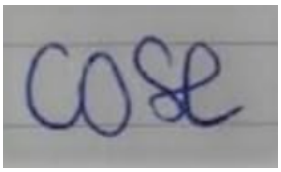
Questi modelli sono stati testati su varie immagini contenenti testo e sono successivamente stati confrontati i risultati

Figura 3.6: Immagine di input - perché



Modello	Output
Keras	perche
Rosetta	perche
Tesseract	perché
Simple HTR	pERCHE |

Figura 3.7: Immagine di input - cose



Modello	Output
Keras	woe
Rosetta	cose
Tesseract	cose
Simple HTR	| cose |

Figura 3.8: Immagine di input - teoria degli insiemi

Modello	Output
Keras	teoria deali insiens
Rosetta	teoria degut insiemt
Tesseract	TEORIA DEGLI INSIEMI
Simple HTR	TEORIA DeGu wSIEml

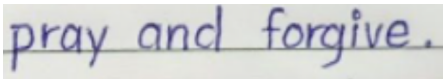
Figura 3.9: Immagine di input - Io, un.

Modello	Output
Keras	105 un
Rosetta	iuun
Tesseract	io, un.
Simple HTR	i, n.

Figura 3.10: Immagine di input - L'Antologia Palatina, una raccolta di epigrammi.

Modello	Output
Keras	di l'antologia palatgina raccolta epigrammis una
Rosetta	vantologia palatina una raccolta di epigrammis
Tesseract	l'Antologia Palatina, una raccolta di epigrammi.
Simple HTR	IIntologia Palatinas una raccoleadiepigrammin

Figura 3.11: Immagine di input - Pray and forgive.



Modello	Output
Keras	and forgive pray
Rosetta	pray and forgive
Tesseract	pray and forgive
Simple HTR	prayandgrien

Come si può chiaramente vedere dai test effettuati il modello che dà i migliori risultati è sicuramente Tesseract.

3.2.6.3 TesserOCR

Si tratta di un semplice wrapper Python della libreria Tesseract per il riconoscimento ottico dei caratteri (OCR), di cui abbiamo parlato precedentemente. [6, 28]
TesserOCR si integra direttamente con le API C++ di Tesseract utilizzando Cython, ciò consente un codice sorgente Python semplice e di facile lettura; inoltre permette anche l'esecuzione concorrente reale, quando usato in concomitanza al modulo di threading di Python. [6]

TesserOCR può essere usato in concomitanza con **Pillow**[4], per la gestione delle immagini dalle quali ricava il testo, ma può comunque essere usato direttamente con i file delle immagini. [6]

Di seguito uno snippet di codice Python all'interno del quale viene utilizzato PIL per aprire un'immagine dalla quale viene estrapolato il testo con l'OCR, tale testo viene poi stampato sulla console (Code Listing: 3.7).

```
1 import tesserocr
2 from PIL import Image
3
4 image = Image.open('sample.jpg')
5 print(tesserocr.image_to_text(image))
```

Code Listing 3.7: Esempio di uso di PIL e TesserOCR

Questo wrapper Python è inoltre l'unico, che permette il riconoscimento del **font** e di altri **attributi dei caratteri** grazie alla funzione WordFontAttributes() che ritorna un dizionario composto da:

- **font name** –> stringa che rappresenta il nome del font utilizzato
- **bold** –> un valore booleano, True quando la parola è in grassetto
- **italic** –> un valore booleano, True quando la parola è in corsivo
- **monospace** –> un valore booleano, True quando le lettere della parola sono di grandezza fissa
- **serif** –> un valore booleano, True quando il font della parola ha le grazie
- **pointsize** –> la larghezza e l'altezza di un carattere misurate in punti
- **font id** –> un intero rappresentante l'identificatore univoco del font

Di seguito una sezione di codice Python all'interno della quale viene utilizzato TesserOCR per estrapolare le informazioni sul font e gli attributi dei caratteri del testo contenuto nell'immagine "image" (Code Listing: 3.8).

```
1 SetImage(image)
2 Recognize()
3 for word in iterate_level(api.GetIterator(), RIL.WORD):
4     print(word.WordFontAttributes())
```

Code Listing 3.8: Esempio di uso di TesserOCR per l'estrapolazione degli attributi dei carattery

3.3 Tecnologie web

All'interno di questa sezione verranno trattate le principali tecnologie web utilizzate all'interno del progetto; la sezione si divide in tre principali sottosezioni. La prima sezione è dedicata all'**HTML** (HyperText Markup Language - linguaggio a marcatori per ipertesti) il linguaggio standard per i documenti progettati per essere visualizzati in un browser web. [17]
Le sezioni successive sono popolare dalle tecnologie che assistono l'HTML all'interno delle pagine web come il **CSS** (Cascading Style Sheets - fogli di stile a cascata) e il linguaggio di scripting **JavaScript**, utilizzato per creare effetti dinamici e interattivi attraverso l'uso di funzioni invocate da eventi. [16, 18]

3.3.1 HTML

L'HTML è un linguaggio di MarkUp di pubblico dominio, la cui sintassi è stabilita dal W3C (World Wide Web Consortium).
Gli elementi costitutivi delle pagine HTML sono i tag HTML delineati utilizzando le parentesi angolari.
Di seguito la copia di un documento HTML di esempio, con il quale viene rappresentata una pagina contenente il testo "Hello world!" (Code Listing: 3.9).

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Il titolo della pagina</title>
5    </head>
6    <body>
7      <div>
8          <p>Hello world!</p>
9      </div>
10   </body>
11 </html>
```

Code Listing 3.9: Esempio di HTML

Ogni pagina HTML deve essere composta dall'elemento **radice HTML** che deve necessariamente contenere gli elementi body e head.
Il **tag head** è il contenitore dei metadati riguardanti l'HTML e per questo non vengono visualizzati all'interno della pagina. Gli elementi, che più comunemente si trovano nel tag head sono:

- **title** –> questo tag obbligatorio definisce il titolo del documento, che deve essere di solo testo e viene mostrato nella barra del titolo del browser o nella scheda della pagina (Code Listing: 3.10)

```
1 <title>Il titolo della pagina</title>
```

Code Listing 3.10: Esempio di definizione di titolo

Il contenuto del titolo di una pagina è molto importante per l'**ottimizzazione dei motori di ricerca**, perché viene utilizzato dagli algoritmi per decide-

re l'ordine delle pagine nei risultati di ricerca, quindi deve essere il più accurato e significativo possibile

- **style** –> questo tag viene utilizzato per definire le informazioni di stile del documento grazie al CSS (Code Listing: 3.11)

```
<style>
  h1 {color: red;}
  p {color: blue;}
</style>
```

Code Listing 3.11: Esempio di definizione di stile

lo stile può anche essere preso da un foglio di stile esterno grazie al tag link (Code Listing: 3.12)

```
            <link rel="stylesheet" href="stile_file.css">
```

Code Listing 3.12: Esempio di definizione di link

- **script** –> questo tag viene utilizzato per incorporare uno script lato client, tale script può essere contenuto direttamente all'interno del tag (Code Listing: 3.14)

```
<script  type="text/javascript">
    alert(Hello world!)
</script>
```

Code Listing 3.13: Esempio di definizione di script

oppure può rimandare a uno script esterno tramite l'attributo "src" (Code Listing: 3.14)

```
<script src="script_file.js"></script>
```

Code Listing 3.14: Esempio di definizione di script con src

- **meta** –> i tag meta servono a specificare determinati metadati, che non vengono visualizzati dalla pagina, ma sono machine parsable e vengono utilizzati dai browser, i motori di ricerca e altri servizi web (Code Listing: 3.15)

```
<meta charset="UTF-8">
<meta name="description" content="Una descrizione della
   pagina">
```

```
3 <meta name="keywords" content="Le, keyword, della, pagina">
4 <meta name="author" content="L'autore della pagina">
```

Code Listing 3.15: Esempio di definizione di meta tag

mentre il tag meta viewport consente ai web designer di assumere il controllo dell'area visibile dall'utente che visita la pagina (Code Listing: 3.16)

```
1 <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
```

Code Listing 3.16: Esempio di definizione del meta tag viewport

Il tag **body** definisce il corpo del documento, contiene tutti i contenuti di un documento HTML; può essercene solamente uno all'interno della pagina. Gli elementi, che più comunemente si trovano nel body, sono:

- **p** –> questo tag serve a definire un paragrafo, i browser aggiungono automaticamente una riga vuota prima e dopo l'elemento (Code Listing: 3.17)

```
1 <p>Hello world!</p>
```

Code Listing 3.17: Esempio di definizione di p

- **br** –> questo tag serve per inserire interruzioni di riga (Code Listing: 3.18)

```
1 <p>
2     Questo paragrafo ha
3     <br>
4     un interruzione di riga
5 </p>
```

Code Listing 3.18: Esempio di definizione di hr

- **h1 - h6** –> questi tag vengono usati per definire le intestazioni HTML, solitamente si usa un solo h1 per pagina e non si salta livelli di intestazione (Code Listing: 3.19)

```
1 <h1>Titolo 1</h1>
2 <h2>Titolo 2</h2>
```

```
3 <h3>Titolo 3</h3>
4 <h4>Titolo 4</h4>
5 <h5>Titolo 5</h5>
6 <h6>Titolo 6</h6>
```

Code Listing 3.19: Esempio di definizione di titoli

- **div** –> questo tag serve a definire una sezione all'interno di un documento, viene utilizzato come contenitore per altri contenitori HTML. I browser impostano sempre una riga prima e dopo l'elemento (Code Listing: 3.20)

```
1 <div>
2     <h1>Titolo della sezione</h1>
3     <p>Questo paragrafo si trova dentro una sezione.</p>
4 </div>
```

Code Listing 3.20: Esempio di definizione di div

- **a** –> questo tag serve a definire un collegamento ipertestuale, utilizzato per collegare una pagina all'altra. L'attributo più importante di questo elemento è href, che indica la destinazione del collegamento (Code Listing: 3.21)

```
1 <a href="destinazione.html">Link</a>
```

Code Listing 3.21: Esempio di definizione di a

L'HTML può quindi incorporare programmi scritti in un linguaggio di scripting come il JavaScript, per influenzare il comportamento e il contenuto delle pagine web. [17]
Nuovi elementi possono essere aggiunti alle pagine grazie all'ausilio di librerie JavaScript come:

3.3.1.1 NicEdit

Si tratta di un WYSIWYG editor (What You See Is What You Get - quello che vedi è quello che è) il cui intento è essere il più veloce e semplice possibile.
L'autore di NicEdit è Brian Kirchoff, che lo ha rilasciato nel 2007-2008 sotto licenza MIT.
NicEdit è estremamente leggero e può essere facilmente integrato in qualsiasi

sito con un impatto minimo, fornendo ai visitatori un mezzo efficace per utilizzare del rich text.
L'editor è completamente configurabile durante il processo di creazione, possono essere modificati:

- cambiare percorso ai file delle icone (Code Listing: 3.22)

```
1 new nicEditor({iconsPath : 'nicEditorIcons.gif'}).
    panelInstance('area');
```

Code Listing 3.22: Esempio cambio icone per NicEdit

- aggiunta o rimozione e ordine dei bottoni (Code Listing: 3.23)

```
1 new nicEditor({buttonList : ['fontSize','bold','italic','
    underline','strikeThrough','subscript','superscript']}).
    panelInstance('area');
```

Code Listing 3.23: Esempio scelta bottoni per NicEdit

- dimensione della textarea (Code Listing: 3.24) [20]

```
1 new nicEditor({maxHeight : 100}).panelInstance('area');
```

Code Listing 3.24: Esempio cambio dimensione per NicEdit

3.3.1.2 SweetAlert2

Si tratta di un'alternativa all'uso dei popup JavaScript completamente personalizzabile e con nessuna dipendenza. SweetAlert 2 è il successore di SweetAlert ed è stato rilasciato per la prima volta nel 2015 sotto licenza MIT.
Dopo aver importato i file all'interno dell'applicazione si potranno utilizzare i popup semplicemente richiamando il metodo 'fire' della classe 'Swal', dopo il completo caricamento del DOM.

Di seguito lo snippet di codice necessario a richiamare un alert SweetAlert contenente la stringa "Hello World!" (Code Listing: 3.25).

```
1 Swal.fire('Hello world!');
```

Code Listing 3.25: Esempio di uso di SweetAlert

Il metodo 'fire' può avere molti parametri, i principali sono:

- **title, text** –> due stringhe che rappresentano il titolo visualizzato in alto e sotto una descrizione (Code Listing: 3.26)

```
1 Swal.fire({
2   title: 'Il titolo',
3   text: 'La descrizione'
4 })
```

Code Listing 3.26: Esempio titolo e testo per weetAlert

- **icon** –> delle icone animate identificabili attraverso una stringa come: 'error', 'info', 'success', ... (Code Listing: 3.27)

```
1 Swal.fire({
2   icon: 'info',
3   title: 'Il titolo',
4   text: 'La descrizione'
5 })
```

Code Listing 3.27: Esempio aggiunta icona per weetAlert

- **image** –> esistono diversi parametri per inserire delle immagini all'interno del popup (Code Listing: 3.28)

```
1 Swal.fire({
2   imageUrl: 'https://image.url/...',
3   imageHeight: 1500,
4   imageAlt: 'Image description'
5 })
```

Code Listing 3.28: Esempio uso di immagine per weetAlert

- **HTML** –> questo parametro dà la possibilità di inserire tag HTML all'interno del popup (Code Listing: 3.29) [27]

```
1 Swal.fire({
2   title: 'titolo',
3   html:
4     'Possono essere utilizzati i tag HTML come ' +
```

```
5       '<a> i links</a> '
6   })
```

Code Listing 3.29: Esempio uso di immagine per weetAlert

3.3.2 CSS

Il CSS (cascading style sheet - fogli di stile a cascata) è una tecnologia fondamentale del World Wide Web, insieme a HTML e JavaScript. Si tratta di un **linguaggio per fogli di stile** utilizzato per descrivere la presentazione di un documento scritto in un linguaggio di markup come HTML, XHTML, o XML. Il CSS è progettato per consentire la separazione tra presentazione e contenuto, in modo da migliorare la flessibilità e il controllo nella specificazione delle caratteristiche di presentazione. [16, 17, 18]

Le regole per comporre il CSS sono mantenute e emanate dal W3C.
Le regole nei fogli di stile sono costituite da uno o più selettori, che indicano a quali elementi assumeranno le proprietà e i relativi valori enunciati di seguito. Tali regole prendono la forma (Code Listing: 3.30)

```
1 selettore {
2   proprieta: valore;
3 }
```

Code Listing 3.30: Esempio uso di CSS

Esistono vari tipi di selettori, fra cui:

- **tutti gli elementi** –> (Code Listing: 3.31)

```
1 * {
2   proprieta: valore;
3 }
```

Code Listing 3.31: Esempio uso di selettore * CSS

- **tipologia di elemento** –> (Code Listing: 3.32)

```
1 p {
2   proprieta: valore;
```

```
3 }
```

Code Listing 3.32: Esempio uso di selettore per tipologia di elemento in CSS

- **classe** –> (Code Listing: 3.33)

```
1 .classe {
2   proprieta: valore;
3 }
```

Code Listing 3.33: Esempio uso di selettore per classe di elemento in CSS

- **ID** –> (Code Listing: 3.34)

```
1 #id {
2   proprieta: valore;
3 }
```

Code Listing 3.34: Esempio uso di selettore per id elemento in CSS

- **l'elemento solo quando il cursore ci passa sopra** –> (Code Listing: 3.35) [16]

```
1 elemento::hover {
2   proprieta: valore;
3 }
```

Code Listing 3.35: Esempio uso di selettore per id elemento in CSS

3.3.2.1 Bootstrap

Bootstrap è un **framework per front-end**, più nel dettaglio si tratta di una raccolta di strumenti per la creazione di applicazioni per il Web, che contiene **template CSS** e in alcuni casi, il **codice JavaScript** di accompagnamento, supportato dalle ultime versioni di tutti i principali browser. [15, 31]

Bootstrap è stato sviluppato da Mark Otto e Jacob Thornton per incoraggiare la coerenza tra gli strumenti interni a Twitter, perché l'azienda utilizzava un'ampia varietà di librerie per lo sviluppo, che portava a incongruenze e alti costi di manutenzione. È stato usato per la prima volta durante la prima Hackweek di Twitter, con il nome di **Twitter Blueprint**.

È stato rinominato Bootstrap e rilasciato come progetto **open source**, sotto licenza MIT (Apache 2.0), il 19 agosto 2011. Nel febbraio 2012 è stato il progetto di sviluppo che sulla piattaforma GitHub ha ricevuto il maggior numero di apprezzamenti.
A partire dalla **versione 2.0** supporta anche il **responsive web design**, ciò significa che la struttura grafica dei siti è dinamica e tiene conto delle proprietà del dispositivo utilizzato (PC, tablet, cellulare, ...), gli elementi vengono automaticamente adattati alla risoluzione dello schermo o alla dimensione della finestra.
Fino alla **versione 5** i componenti JavaScript di Bootstrap si basavano sul framework JavaScript JQuery, ma dalla versione 5 la dipendenza è stata eliminata, visto che i browser web principali supportano oggi la maggior parte delle funzionalità di JQuery in **JavaScript Vanilla**; ciò è stato però possibile solamente abbandonando il supporto ad Internet Explorer. [13]
Questa sostituzione ha fatto sì che le applicazioni, che usano la versione 5 di Bootstrap siano più leggere e i tempi di caricamento siano nettamente inferiori, rispetto alle versioni precedenti. [31]

Bootstrap fornisce uno stile per tutti gli **elementi HTML**, in modo da dare un aspetto uniforme alle pagine web; tale stile può anche essere personalizzato dallo sviluppatore, nel CSS, oppure grazie alle classi Bootstrap. [15]
Di seguito alcuni tag HTML con le classi "btn-primary", "btn-light" e "btn-dark" per la decisione del colore dell'elemento (Code Listing: 3.36)

```
1 <button type="button" class="btn btn-primary">Colore primario</
    button>
2 <button type="button" class="btn btn-light">Bottone chiaro</
    button>
3 <button type="button" class="btn btn-dark">Bottone scuro</button>
```

Code Listing 3.36: Esempio uso di Bootstrap

Il framework fornisce, oltre ai normali elementi HTML, **altri elementi**, realizzati da Bootstrap utilizzando tag HTML, usati frequentemente nelle interfacce web. Fra questi elementi troviamo:

- **Alert** –> si tratta di un meccanismo per fornire messaggi di feedback contestuali con messaggi personalizzabili (Code Listing: 3.37)

```
1 <div class="alert alert-primary" role="alert">
2   Messaggio di alert.
3 </div>
```

Code Listing 3.37: Esempio uso di alert Bootstrap

- **Breadcrumb** –> letteralmente "molliche di pane", si tratta dei percorsi di navigazione utilizzati per indicare la posizione della pagina corrente all'interno della gerarchia di navigazione (Code Listing: 3.38)

```
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item">
        <a href="#">Pagina precedente</a>
    </li>
    <li class="breadcrumb-item active" aria-current="page">
        Pagina corrente
    </li>
  </ol>
</nav>
```

Code Listing 3.38: Esempio uso di breadcrumb Bootstrap

- **Accordion** –> letteralmente "fisarmoniche", si tratta di pannelli che collassano verticalmente e possono contenere elementi al loro interno una volta aperti. Funziona in combinazione con il plugin JavaScript di Bootstrap chiamato Collapse (Code Listing: 3.39)

```
<div class="accordion" id="accordionPanelsStayOpenExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="panelsStayOpen-
       headingOne">
      <button class="accordion-button" type="button" data-bs
         -toggle="collapse" data-bs-target="#panelsStayOpen-
         collapseOne" aria-expanded="true" aria-controls="
         panelsStayOpen-collapseOne">
        Pannello 1
      </button>
    </h2>
```

```
8       <div id="panelsStayOpen-collapseOne" class="accordion-
            collapse collapse show" aria-labelledby="
            panelsStayOpen-headingOne">
9         <div class="accordion-body">
10          Contenuto del pannello 1
11        </div>
12      </div>
13    </div>
14    <div class="accordion-item">
15      <h2 class="accordion-header" id="panelsStayOpen-
            headingTwo">
16        <button class="accordion-button collapsed" type="
              button" data-bs-toggle="collapse" data-bs-target="#
              panelsStayOpen-collapseTwo" aria-expanded="false"
              aria-controls="panelsStayOpen-collapseTwo">
17          Pannello 2
18        </button>
19      </h2>
20      <div id="panelsStayOpen-collapseTwo" class="accordion-
            collapse collapse" aria-labelledby="panelsStayOpen-
            headingTwo">
21        <div class="accordion-body">
22          Contenuto pannello 2
23        </div>
24      </div>
25    </div>
26  </div>
```

Code Listing 3.39: Esempio uso di accoridion Bootstrap

3.3.3 JavaScript

JavaScript, solitamente abbreviato in JS, è un linguaggio di programmazione di alto livello orientato agli oggetti e agli eventi.

Viene comunemente usato negli applicativi web per la creazione di **effetti dinamici**, **interattivi** grazie a funzioni di script invocate da eventi innescati dall'utente sulla pagina web in uso. [18, 35]

Di seguito uno snippet di codice JavaScript in cui si stampa in console la stringa "Hello World!". (Code Listing: 3.40)

```
1 function helloWorld() {
2     console.log("Hello World!");
3 }
4 helloWorld();
```

Code Listing 3.40: Esempio di JavaScript

Insieme a HTML e CSS, il JavaScript è una delle tecnologie fondamentali del World Wide Web, tutti i principali browser hanno un motore JavaScript dedicato per eseguire **lato client** il codice.
I motori JavaScript inizialmente utilizzati solo nei browser web, sono oggi le componenti principali di alcuni **server**, il sistema runtime più popolare è Node.js. [17, 16, 18, 35]

Nel 1995 **Netscape** decise di aggiungere un linguaggio di scripting a Navigator, il browser più utilizzato dell'epoca, inizialmente pensarono di collaborare con Sun Microsystem per incorporare il linguaggio di programmazione **Java** e con Brendan Eich per incorporare il linguaggio **Scheme**.
Decisero infine che la cosa migliore fosse ideare un nuovo linguaggio di programmazione con una sintassi simile al Java, ma non a Scheme, per questo sebbene il linguaggio fosse inizialmente chiamato **LiveScript** nel dicembre del 1995 per la versione ufficiale fu cambiato in **JavaScript**. Il nome fu cambiato inoltre come strategia di marketing perché già al tempo Java era un linguaggio molto conosciuto. [35]

Gli script JavaScript utilizzati all'interno delle pagine web devono essere inseriti all'interno delle pagine HTML con il **tag script**, che può contenere direttamente il codice all'interno della pagina HTML stessa (Code Listing: 3.41)

```
1 <script type="text/javascript">
2   alert("Hello world!");
3 </script>
```

Code Listing 3.41: Esempio di tag Script che contiene codice JavaScript

oppure può rimandare a un file contenente codice JavaScript grazie all'attributo 'src' (Code Listing: 3.42) [18]

```
<script type="text/javascript" src="file.js"></script>
```

Code Listing 3.42: Esempio di tag Script che contiene codice JavaScript

3.3.3.1 Vue.js

Vue (pronunciato come view) è un **framework frontend**, open source per la creazione di interfacce utente e applicazioni single-page.
A differenza di altri framework monolitici, VUE, presenta un'architettura adottabile in modo incrementale, che si concentra sul **rendering dichiarativo**. La sua libreria principale è focalizzata solo sul livello di visualizzazione ed è facile da integrare con altre librerie e progetti esistenti, mentre le funzionalità avanzate, come routing e state management sono disponibili tramite librerie di supporto, come Vue Router, router ufficiale di Vue.js o, Vue Server Render, per il rendering lato server. [43, 42]

Vue è stato creato da Evan You in TypeScript e rilasciato sotto licenza MIT nel febbraio del 2014 dopo aver lavorato a un progetto per la Google con AngularJS, anch'esso un framework per applicazioni web, ma molto più pesante. [42]

Al centro di Vue.js c'è un sistema che ci consente di eseguire il **rendering dichiarativo dei dati nel DOM** utilizzando la sintassi dei template. [43]

Di seguito un esempio di uso di Vue per la visualizzazione di una linea di testo in una pagina web (Code Listing: 3.43) scelto attraverso JavaScript (Code Listing: 3.44)

```
<div id="app">
  {{ message }}
</div>
```

Code Listing 3.43: Esempio di uso di Vue - HTML

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!'
```

```
5   }
6 })
```

Code Listing 3.44: Esempio di uso di Vue - JavScript

Per modificare gli elementi HTML collegati all'app VUE, non serve più interagire direttamente con l'HTML, perché VUE li controlla completamente, per questo se all'interno del codice js venisse successivamente cambiato il valore di app.message, il DOM cambierebbe di conseguenza.
Vue offre inoltre un sistema di **effetti di transizione** che può applicare automaticamente quanto gli elementi vengono inseriti/aggiornati/rimossi da Vue. [43]

Oltre a collegare il valore dei tag HTML, possono anche essere collegati gli **attributi** (Code Listing: 3.45)

```
1 <div id="app">
2   <span v-bind:title="message"> L'elemento HTML </span>
3 </div>
```

Code Listing 3.45: Esempio di uso di Vue - HTML

l'attributo "v-bind" è chiamato direttiva, le direttive sono precedute "v-" per indicare che sono attributi speciali forniti da Vue e che applicano un comportamento speciale al DOM, mantenendo aggiornato l'attributo a cui sono affiancate.
Questi attributi speciali possono essere utilizzati anche per:

- **condizioni** –> grazie all'attributo v-if si può alternare la presenza di un elemento HTML (Code Listing: 3.46 - 3.47)

```
1 <div id="app">
2   <span v-if="seen">Visibile</span>
3 </div>
```

Code Listing 3.46: Esempio di uso di if per Vue - HTML

```
1 var app = new Vue({
2   el: '#app',
3   data: {
```

```
4       seen: true
5     }
6   })
```

Code Listing 3.47: Esempio di uso di if per Vue - JavaScript

- **loop** –> la direttiva "v-for" può essere utilizzata per visualizzare elenchi di elementi utilizzando per esempio i dati di un Array (Code Listing: 3.48 - 3.49)

```
1   <div id="app">
2     <ol>
3       <li v-for="element in array">
4         {{ element.text }}
5       </li>
6     </ol>
7   </div>
```

Code Listing 3.48: Esempio di uso di loop per Vue - HTML

```
1   var app = new Vue({
2     el: '#app',
3     data: {
4       array: [
5         { text: 'Testo elemento 1' },
6         { text: 'Testo elemento 2' },
7         { text: 'Testo elemento 3' }
8       ]
9     }
10  })
```

Code Listing 3.49: Esempio di uso di loop per Vue - JavaScript

- **gestire input utente** –> grazie alla direttiva "v-on" si può collegare event listener che invocano metodi dell'istanza Vue (Code Listing: 3.50 - 3.51)

```
1   <div id="app">
2     <p>{{ message }}</p>
```

```
3   <button v-on:click="completeMessage">Complete Message</
        button>
4 </div>
```

Code Listing 3.50: Esempio di uso di v-on per Vue - HTML

```
1  var app = new Vue({
2    el: '#app',
3    data: {
4      message: 'Hello '
5    },
6    methods: {
7      completeMessage: function () {
8        this.message = this.message + 'World!'
9      }
10   }
11 })
```

Code Listing 3.51: Esempio di uso di v-on per Vue - JavaScript

Vue fornisce inoltre la direttiva "v-model" che rende il legame tra input del modulo e stato dell'app bidirezionale (Code Listing: 3.52 - 3.53)

```
1 <div id="app">
2   <p>{{ message }}</p>
3   <input v-model="message">
4 </div>
```

Code Listing 3.52: Esempio di uso di v-model per Vue - HTML

```
1  var app = new Vue({
2    el: '#app',
3    data: {
4      message: 'Hello world!'
5    }
6  })
```

Code Listing 3.53: Esempio di uso di v-model per Vue - JavaScript

3.4 TEI Guidelines

La Text Encoding Initiative (TEI) è un consorzio di ambito linguistico, che ha sviluppato una serie di **linee guida per la codifica di testi umanistici** in formato digitale, per il loro uso da parte di istituzioni, comunità e singoli individui. Queste linee guida sono espresse attraverso uno schema XML modulare ed estensibile, accompagnate da documentazione dettagliata e pubblicate con licenza open source. [5]

Le linee guida sono progettate per essere **personalizzate** a seconda delle particolare esigenze di progetti specifici, come per esempio la popolare versione TEI Lite.

La prima versione ufficiale delle guidelines, la **P3**, venne rilasciata nel 1994 ed era scritta per SGML, solamente a partire dalla versione successiva venne aggiunta la compatibilità con l'XML, la **P4** ed è stata rilasciata nel 2002.
La versione attualmente in uso è la **P5** ed è stata rilasciata nel 2007 e da allora è stato aggiornato ogni sei mesi con correzioni e miglioramenti minori delle funzionalità.

Il formato è usato da molti progetti in tutto il mondo, fra i più importanti ricordiamo:

- **British National Corpus** (**BNC**) –> corpus testuale di cento milioni di campioni dall'inglese britannico della fine del ventesimo secolo scritto e parlato
- **Oxford Text Archive** (**OTA**) –> archivio di testi elettronici e altre risorse letterarie e linguistiche in più di 25 lingue, che sono state create, raccolte e distribuite a scopo di ricerca su argomenti letterari e linguistici
- **Perseus Project** –> progetto di biblioteca digitale della Tufts University, che riunisce raccolte digitali di risorse umanistiche in greco e latino
- **Women Writers Project** (**WWP**) –> progetto di ricerca e pubblicazione che si concentra sulla messa a disposizione online dei teti delle prime scrittrici moderne in lingua inglese [41]

3.4.1 Struttura del testo

Un documento TEI P5 minimo è formato da:

- **prologo XML** e **DOCTYPE**, necessari in tutti i documenti XML (Code Listing: 3.54)

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE TEI SYSTEM "tei.dtd">
```

Code Listing 3.54: Esempio di prologo XML e DOCTYPE

- l'elemento **radice TEI** (Code Listing: 3.55)

```
1 <TEI xmlns="http://www.teic.org/ns/1.0">
```

Code Listing 3.55: Esempio di radice TEI

- il TEI header, o intestazione TEI contenente metadati relativi al documento (Code Listing: 3.56)

```
1 <teiHeader>
2     ...
3 </teiHeader>
```

Code Listing 3.56: Esempio di TEI Header

- gli elementi strutturali sono quelli che servono a rappresentare il testo vero e proprio. Ne esistono vari, per esempio (Code Listing: 3.57)

```
1 <text> ... </text>
2 <facsimile> .. </facsimile>
```

Code Listing 3.57: Esempio di text e facsimile TEI

3.4.1.1 TEI Header

Il TEI header è un tag necessario per il corretto funzionamento di un XML:TEI e contiene i metadati relativi al documento come la descrizione del file. Gli elementi essenziali sono quelli riguardanti:

- il **titolo** –> sezione dedicata alla codifica del titolo dell'edizione digitale (Code Listing: 3.58)

```
1 <titleStmt>
2     <title>Titolo opera</title>
3     <respStmt>
4         <resp>Conversione TEI a cura di</resp>
```

```
5         <name>Deep Text Recognition</name>
6     </respStmt>
7 </titleStmt>
```

Code Listing 3.58: Esempio di titolo TEI

- la **modalità di diffusione** –> sezione dedicata alla codifica dei dati di diffusione e distribuzione dell'edizione (Code Listing: 3.59)

```
1 <publicationStmt>
2     <publisher>Editore</publisher>
3     <date>YYYY-MM-DD</date>
4     <availability>Disponibilita opera</availability>
5 </publicationStmt>
```

Code Listing 3.59: Esempio di definizione della modalità di diffusione TEI

- la **fonte originaria** –> sezione dedicata alla codifica dei dati relativi alla fonte primaria (Code Listing: 3.60)

```
1 <sourceDesc>
2     <bibl>
3         <title>Il titolo della fonte</title>
4         <author>Autore della fonte</author>
5         <publisher>Editore della fonte</publisher>
6         <date>YYYY-MM-DD</date>
7     </bibl>
8 </sourceDesc>
```

Code Listing 3.60: Esempio di descrizione della fonte TEI

Esistono anche altri elementi, che possono essere inseriti all'interno dell'intestazione come:

- informazioni riguardanti lo schema e il modello di codifica utilizzato (Code Listing: 3.61)

```
1 <encodingDesc>
2     ...
3 </encodingDesc>
```

Code Listing 3.61: Esempio di definizione del tag contenente le informazioni di codifica TEI

- descrizione del testo come quando è stato creato, da chi, in quali lingue, ... (Code Listing: 3.62)

```
<profileDesc>
    ...
</profileDesc>
```

Code Listing 3.62: Esempio di definizione del tag contenente le informazioni sul testo TEI

- informazioni sulle versioni del file (Code Listing: 3.63)

```
<revisionDesc>
    ...
</revisionDesc>
```

Code Listing 3.63: Esempio di definizione del tag contenente le informazioni sulle versioni TEI

3.4.1.2 Elementi strutturali

Come sottolineato precedentemente gli elementi strutturali sono quelli che vanno a rappresentare il testo del documento.
Esistono vari elementi strutturali i principali sono:

3.4.1.2.1 Text L'elemento strutturale **text** serve a rappresentare un singolo testo di qualsiasi tipo e può contenere:

- **front** –> racchiude tutti i contenuti che precedono il corpo del testo
- **body** –> viene usato per rappresentare il testo stesso, a sua volta contiene una struttura gerarchica ad albero del documento con ulteriori divisioni (paragrafi, capitoli, ...)
- **front** –> contiene tutti i materiali che possono seguire il corpo del testo, come per esempio le appendici e gli annessi (Code Listing: 3.64)

```
<text>
    <front> ... </front>
    <body> ... </body>
    <back> ... </back>
</text>
```

Code Listing 3.64: Esempio di definizione del tag text TEI

- **group** –> alternativa a body utilizzato per raggruppare testi diversi, in alternativa per rappresentare testi compositi può anche essere usato il tag group al posto del tag text, che può poi andare a contenere più tag text (Code Listing: 3.65)

```
<text>
    <group>
        <text>
            <front> ... </front>
            <body> ... </body>
            <back> ... </back>
        </text>
    <group>
    <group>
        <text>
            <front> ... </front>
            <body> ... </body>
            <back> ... </back>
        </text>
    <group>
</text>
```

Code Listing 3.65: Esempio di definizione del tag group TEI

Moltissimi altri tag possono essere usati all'interno del testo per rappresentarne varie caratteristiche, per esempio:

- **div** –> vengono utilizzati per suddividere il testo in più sezioni e non hanno nessun limite di nidificazione i suoi attributi principali sono type,

per descrivere di quale tipologia di sezione si tratta e n per dare una consequenzialità alle sezioni (Code Listing: 3.66)

```
<div type="tipologia_di_sezione" n="1">
    ...
</div>
<div type="tipologia_di_sezione" n="2">
    ...
</div>
```

Code Listing 3.66: Esempio di definizione del tag div TEI

- **p** –> tag utilizzato, come in HTML, per rappresentare i paragrafi (Code Listing: 3.67)

```
<p> ... </p>
```

Code Listing 3.67: Esempio di definizione del tag p TEI

- **emph** –> viene usato per parole o frasi enfatizzate all'interno del testo, grazie all'attributo rend si può anche descrivere come tale sezione di testo sia enfatizzata con Bold, Italic, ... (Code Listing: 3.68)

```
<emph rend="italics">testo in corsivo</emph>
<emph rend="bold">testo in grassetto</emph>
<emph rend="underlined">testo sottolineato</emph>
```

Code Listing 3.68: Esempio di definizione del tag emph TEI

- **abbr** –> utilizzato per rappresentare un'abbreviazione assieme al tag expan che ne rappresenta la controparte espansa dell'abbreviazione (Code Listing: 3.69)

```
<choice>
    <abbr>abreviazione</abbr>
    <expan>versione espansa abbreviazione</expan>
</choice>
```

Code Listing 3.69: Esempio di definizione del tag abbr TEI

3.4.1.2.2 Facsimile L'elemento facsimile serve alla riproduzione della fonte primaria, può affiancare o sostituire l'elemento Text.
Il suo scopo è quello di fornire un modo per visualizzare parallelamente il testo codificato e la rappresentazione digitalizzata del documento originale. Il modo più comune per realizzare una visualizzazione parallela di immagini del testo e testo codificato è rappresentata in Code Listing: 3.70

```
<facsimile>
    <surface>
        <graphic url="immagine_della_pagina.jpg"/>
        <zone points = ' 00, 00 00,00 00,00 00,00 ' >
        <figure>
            <graphic url="immagine_del_dettaglio1" />
            <note> descrizione del dettaglio 1 </note>
        </figure>
        <zone points = ' 00, 00 00,00 00,00 00,00 ' >
        <figure>
            <graphic url="immagine_del_dettaglio2" />
            <note> descrizione del dettaglio 2 </note>
        </figure>
    </surface>
    <surface>
        ...
    </surface>
    ...
</facsimile>
```

Code Listing 3.70: Esempio di visualizzazione parallela di immagini e testo con TEI

dove:

- **surface** –> utilizzato per indicare una superficie scritta, che raggruppi eventualmente una o più rappresentazioni grafiche di interesse di tale spazio e delle zone all'interno di quest'ultimo
- **graphic** –> utilizzato per indicare il path di un grafico, di un'illustrazione o di un'immagine grazie all'attributo url

- **zone** –> definisce uno spazio bidimensionale all'interno di un elemento superficie, si passano le coordinate di tale zona tramite l'attributo point, o tramite gli attributi ulx, uly, lrx, lry
- **note** –> utilizzato per scrivere dei commenti sui particolari rappresentati dal graphic soprastante

3.4.2 Infrastruttura modulare

Il sistema TEI è un sistema modulare, grazie al quale è possibile selezionare solamente i moduli necessari, in modo da realizzare velocemente uno schema di codifica appropriato.
Ogni **modulo** contiene un certo numero di **elementi** detti tagset, tali elementi e i loro attributi sono a loro volta organizzati in classi.
I **moduli** TEI di **base** sono: TEI, header, textstructure e core; anche usando solamente i moduli essenziali si ha a disposizione uno schema adatto alla marcatura di un gran numero di testi.

Illustrati di seguito i principali moduli TEI.

3.4.2.1 Analysis

Questo modulo contiene strumenti per l'analisi e l'interpretazione, semantica o sintattica, del testo. Esistono varie tipologie di elementi:

- **segmenti linguistici** –> utilizzata per caratterizzare segmenti di testo secondo le categorie linguistiche familiari di **frase**, **parola**... (Code Listing: 3.71)

```
1 <s>frase complessa</s>
2 <cl>parte della frase con soggetto e verbo</cl>
3 <phr>parte della frase che NON contiene soggetto e verbo</
     phr>
4 <w>parola</w>
```

Code Listing 3.71: Esempio di segmenti linguistici con TEI

- **elementi interpretativi** –> i principali elementi di questo gruppo servono per associare un'annotazione interpretativa direttamente a un intervallo di testo (**span**), oppure a riassumerne una che può poi essere collegata a un intervallo di testo (**interp**) (Code Listing: 3.72)

```
<span> ... </span>
<interp> ... </interp>
```

Code Listing 3.72: Esempio di elementi interpretativi con TEI

e l'attributo **@ana** per l'analisi, che può essere specificato per qualsiasi elemento.

- **annotazioni linguistiche** –> per annotazioni linguistiche si intende una qualsiasi annotazione determinata da un'analisi delle caratteristiche linguistiche del testo, escludendo come casi limite le sue proprietà strutturali e le sue informazioni descrittive del contesto, che possono essere annotati tramite l'uso di tag presenti in altri moduli. I principali attributi sono **@lemma** che serve sia come identificatore, sia come base per potenziali inflessioni, **@pos**, (part of speech - parte del discorso) che indica la parte di un discorso assegnata a un token e **@msd** (morphosyntactic description - descrizione morfosintattica) fornisce informazioni morfosintattiche per un token, solitamente secondo un vocabolario di riferimento ufficiale (STTS-large tagset per il tedesco) (Code Listing: 3.73)

```
<w pos="..." lemma="...">parola</w>
```

Code Listing 3.73: Esempio di annotazioni linguistiche con TEI

3.4.2.2 Corpora linguistici

Con il termine corpus linguistico si intende una collezione di testi selezionati e organizzati per facilitare determinati criteri di progettazione.
Tali criteri possono essere molto semplici e poco impegnativi, o molto sofisticati e può inoltre essere composto da testi interi oppure da frammenti, o campioni di testo.
A parte queste differenze strutturali e alle possibili differenze di scala, la codifica dei corpora linguistici e dei singoli testi hanno problemi identici.

Per rappresentare un corpus deve essere usato l'elemento **teiCorpus** (TEI corpus), che va a contenere l'intero corpus codificato TEI e uno o più elementi, ciascuno contenente una singola intestazione di testo e un testo (Code Listing: 3.74)

```
<teiCorpus version="3.3.0" xmlns="http://www.tei-c.org/ns/1.0">
    <teiHeader>
```

```
    <!-- intestazione del corpus -->
     </teiHeader>
     <TEI>
        <teiHeader>
            <!-- intestazione del primo testo -->
        </teiHeader>
        <text>
            <!-- contenuto del primo testo -->
        </text>
     </TEI>
     <TEI>
        <teiHeader>
          <!-- intestazione del secondo testo -->
        </teiHeader>
        <text>
            <!-- contenuto del secondo testo -->
        </text>
    </TEI>
    ...
</teiCorpus>
```

Code Listing 3.74: Esempio di teiCorpus TEI

I principali tag che possono essere usati nell'intestazione per la descrizione dei corpus sono

- **text description - descrizione del testo** –> fornisce una descrizione del testo (Code Listing: 3.75)

```
<textDesc>
    <channel mode=""/> <!-- canale di comunicazione -->
    <constitution type="" /> <!-- costituzione del testo -->
    <derivation type=""> <!-- derivazione del testo -->
    ...
</textDesc>
```

Code Listing 3.75: Esempio di descrizione del testo TEI

- **participation description - descrizione della partecipazione** –> descrive le voci partecipanti al testo, o altre persone nominate, o citate (Code Listing: 3.76)

```
<particDesc>
    <listPerson>
        <person>
            <persName>
                <surname>Cognome</surname>
                <forename>Nome</forename>
            </persName>
            <address>
                <street>strada</street>
                <settlement>insediamento</settlement>
            </address>
            ...
        </person>
        <person>
            ...
        </person>
        ...
    </listPerson>
</particDesc>
```

Code Listing 3.76: Esempio di descrizione della partecipazione TEI

- **setting description - descrizione dello scenario** –> descrive lo scenario, o gli scenari nei quali si svolge l'interazione linguistica (Code Listing: 3.77)

```
<settingDesc>
    <p> descrizione... </p>
</settingDesc>
```

Code Listing 3.77: Esempio di descrizione dello scenario TEI

3.4.2.3 Drama

Questo modulo viene è destinato alla rappresentazione della drammaturgia stampata, riprodotta su schermo, radiofonica, o qualsiasi altra forma di performance.

I testi di questa tipologia sono spesso suddivisi in strofe e versi che in TEI vengono rappresentati come in Code Listing: 3.78

```
<lg> <!-- la strofa -->
    <l> ... </l> <!-- i versi -->
    <l> ... </l>
    ...
</lg>
```

Code Listing 3.78: Esempio di drammaturgia TEI

Esistono poi molti elementi per discutere i vari aspetti delle performance, i principali sono:

- **cast** –> grazi a questo tag possono essere rappresentati i partecipanti al cast (Code Listing: 3.79)

```
<castList>
    <castItem>
            <role>ruolo</role>
             <actor>attore</actor>
    </castItem>
    ...
</castList>
```

Code Listing 3.79: Esempio di cast list TEI

- **set** –> questo tag serve a descrivere l'ambientazione o le ambientazioni

```
<set>
    <p>descrizione</p>
</set>
```

- **performance** –> questo tag serve a descrivere come deve essere eseguito una performance, o come è stato eseguito in un'occasione specifica (Code Listing: 3.80)

```
<performance>
    <p>descrizione</p>
    <castList>
        ...
    </castList>

</performance>
<set>
    ...
</set>
```

Code Listing 3.80: Esempio di performance TEI

- **prologue** –> tale elemento serve a contenere il prologo di un dramma, tipicamente pronunciato da un attore, possibilmente in associazione con una particolare performance o luogo (Code Listing: 3.81)

```
<prologue>
    <sp>
        <speaker>chi lo pronuncia</speaker>
        <p> il testo </p>
    </sp>
</prologue>
```

Code Listing 3.81: Esempio di prologo TEI

- **epilogue** –> questo elemento contiene l'epilogo di un dramma tipicamente pronunciato da un attore e possibilmente in associazione con un particolare luogo o performance (Code Listing: 3.82)

```
    <epilogue>
        <sp>
            <speaker>chi lo pronuncia</speaker>
            <p> il testo </p>
        </sp>
    </epilogue>
```

Code Listing 3.82: Esempio di epilogo TEI

3.4.2.4 Gaiji

Questo modulo dà la possibilità di codificare caratteri e glifi non standard e la modalità di scrittura. I glifi o caratteri non standard vengono rappresentati grazia a tag come glyph (glifo) che serve a fornire informazioni descrittive di un glifo (Code Listing: 3.83).

```
<glyph xml:id="rstroke">
    <localProp name="nome del glifo" value="valore del glifo"/>
    <figure>
        <graphic url="immagin_glifo.png"/>
    </figure>
</glyph>
```

Code Listing 3.83: Esempio di glifo TEI

Fra gli attributi utilizzati per specificare la modalità di scrittura, ovvero il modo in cui questi glifi sono disposti sul supporto di scrittura, troviamo invece l'attributo style (Code Listing: 3.84):

```
<lg  style="writing-mode: vertical-rl">
<!-- gruppo di versi scritto in verticale da destra verso
   sinistra-->
     <l>primo verso</l>
     ...
</lg>
<lg  style="writing-mode: upright">
<!-- gruppo di versi scritto orizzontalmente e visualizzato in
   posizione verticale -->
     ...
</lg>
...
```

Code Listing 3.84: Esempio di glifo TEI

3.4.2.5 Manuscripts description

Questo modulo, chiamato msDesc (manuscripts description - descrizione di manoscritto), definisce un elemento omonimo, che può essere utilizzato per definire i metadati relativi alle **fonti primarie** scritte a mano e altri **oggetti contenenti testo**.

Fu inizialmente progettato e sviluppato per soddisfare le esigenze di catalogatori e studiosi che lavoravano con manoscritti medievali di tradizione europea.
Viene però utilizzato per qualsiasi tipo di testo, grazie allo schema abbastanza generale.

I principali tag che possono essere contenuti da **msDesc** sono:

- **msIdentifier** (manuscript identifier - identificativo del manoscritto) –> la funzione di questo tag è quella di contenere le informazioni necessarie all'identificazione del manoscritto (Code Listing: 3.85)

```
<msIdentifier>
    <settlement>insediamento</settlement>
    <repository>deposito</repository>
    <idno>identificativo univoco</idno>
</msIdentifier>
```

Code Listing 3.85: Esempio di manuscript identifier TEI

- **msContents** (manuscript content - contenuto del manoscritto) –> descrive il contenuto intellettuale di un manoscritto (Code Listing: 3.86)

```
<msContents>
    <msItem>
       <author>Autore del manoscritto</author>
       <title>Titolo del manoscritto</title>
       <textLang>Lingua del manoscritto</textLang>
    </msItem>
 </msContents>
```

Code Listing 3.86: Esempio di manuscript content TEI

- **phyDesc** (physical description - descrizione fisica) –> questo tag serve a contenere una descrizione fisica completa del manoscritto (Code Listing: 3.87)

```
<physDesc>
    <objectDesc form="tipologia di oggetto">
        <supportDesc material="materiale oggetto">
            <support>tipologia di supporto di scrittura</
               support>
            <extent>
                <dimensions unit="unita di misura">
                    <height>altezza</height>
                    <width>larghezza</width>
                </dimensions>
            </extent>
        </supportDesc>
        <layoutDesc>
            <layout columns="">in quante colonne</layout>
        </layoutDesc>
    </objectDesc>
    <handDesc>
        <p>descrizione e numero di mani con cui e stato
           scritto il manoscritto</p>
    </handDesc>
    <decoDesc>
        <p>descrizione della decorazione</p>
    </decoDesc>
</physDesc>
```

Code Listing 3.87: Esempio di physical description TEI

3.4.2.6 Spoken

Il modulo spoken (parlato) serve alla descrizione di un'ampia varietà di materiale parlato e trascritto.
La struttura complessiva di un testo parlato è identica a quella di qualsiasi

altro testo TEI, l'elemento TEI contiene l'elemento teiHeader seguito da un elemento testuale.

Il discorso considerato puramente come fenomeno acustico e il discorso considerato esclusivamente come un processo di interazione sociale, possono richiedere metodi diversi rispetto a quelli delineati da questo modulo.

I principali tag per rappresentare il testo trascritto sono:

- **u** (utterance - espressione) –> tag contenente un tratto di discorso solitamente preceduto e seguito da silenzio o da cambio di oratore (Code Listing: 3.88)

```
<u> tratto di discorso </u>
```

Code Listing 3.88: Esempio di utterance TEI

- **pause** (pausa) –> questo elemento segna una pausa tra, o all'interno di enunciati (Code Listing: 3.89)

```
<pause dur="durata"/>
```

Code Listing 3.89: Esempio di pause TEI

- **vocal** (vocale) –> tale tag serve a segnalare qualsiasi fenomeno vocalizzato, ma non necessariamente lessicale, come ad esempio pause sonore, o fischi (Code Listing: 3.90)

```
<vocal iterated="ripetuto (true|false)" dur="durata">
    <desc>descrizione del fenomeno vocale</desc>
</vocal>
```

Code Listing 3.90: Esempio di pause TEI

- **kinesic** (cinesica) –> il tag kinesic serve a contrassegnare qualsiasi fenomeno comunicativo non necessariamente vocalizzato, come un gesto, un cipiglio, ... (Code Listing: 3.91)

```
<kinesic iterated="ripetuto (true|false)" dur="durata">
    <desc>descrizione del fenomeno vocale</desc>
</kinesic>
```

Code Listing 3.91: Esempio di kinesic TEI

- **incident** (incidente) –> questo elemento serve alla rappresentazione di qualsiasi fenomeno o evento, non necessariamente vocalizzato o comunicativo, come dei rumori accidentali, o altri eventi che influenzano la comunicazione (Code Listing: 3.92)

```
<incident>
    <desc>descrizione di un incidente</desc>
</incident>
```

Code Listing 3.92: Esempio di incident TEI

- **writing** (scrittura) –> l'elemento writing contiene dei passaggi di testo scritto mostrato agli ascoltatori del discorso (Code Listing: 3.93)

```
<writing>testo scritto</writing>
```

Code Listing 3.93: Esempio di writing TEI

Esistono inoltre di tag per la documentazione delle fonti del discorso trascritto, che possono essere inseriti all'interno dell'header TEI, i principali sono:

- **scriptStmt** (script statement - dichiarazioni sul copione) –> questo tag contiene informazioni dettagliate sul copione utilizzato per un testo orale (Code Listing: 3.94)

```
<scriptStmt>
    <bibl>
        <author>autore</author>
        <title>titolo</title>
        <edition>edizione</edition>
        ...
    </bibl>
</scriptStmt>
```

Code Listing 3.94: Esempio di script statement TEI

- **recordingStmt** (recording statement - dichiarazioni sulla registrazione) –> tale elemento serve a descrivere un insieme di registrazioni utilizzate come base per la trascrizione di un testo parlato (Code Listing: 3.95)

```
<recordingStmt>
    <recording type="audio" dur="P30M">
```

```
3         <equipment>
4             <p>descrizione della attrezzatura usata</p>
5         </equipment>
6         <date>DD-MM-YYYY</date> <!-- la data di
              registrazione -->
7     </recording>
8 </recordingStmt>
```

Code Listing 3.95: Esempio di recording statement TEI

- **trascriptionDesc** (transcription description - descrizione di trascrizione) –> questo elemento descrive l'insieme delle convenzioni di trascrizione utilizzati (Code Listing: 3.96)

```
1 <transcriptionDesc ident="identificativo" version="versione"
      />
```

Code Listing 3.96: Esempio di transcription description TEI

Capitolo 4

Applicazione

Il progetto oggetto di questa tesi consiste nella progettazione e messa a punto di un applicativo web, realizzato tramite l'uso di Artificial Intelligence per il **riconoscimento di caratteri manoscritti**, all'interno di immagini di **rilevanza umanistica**, e l'elaborazione di quest'ultimi al fine di rendere i dati fruibili in modo agevole e flessibile.
L'utilizzo di un tale sistema potrebbe essere quello della finale codifica in vari formati dei testi riconosciuti. Fra tali codifiche ricordiamo il formato XML:TEI, fondamentale per lo scambio e la conservazione di grandi moli di dati di rilevanza umanistica, tale formato potrebbe essere utile a un utente che decide di creare, o arricchire archivi e/o banche dati testuali.
Si è deciso di chiamare l'applicazione "**Deep Text Recognition**" a ricordarne l'obiettivo e il funzionamento.

Il quarto capitolo si divide in due sezioni:

- all'interno delle sezioni 4.1 vedremo un **analisi dei requisiti** dell'applicazione Deep Text Recognition, realizzata allo scopo di definire le funzionalità del sistema.
  Tale analisi sarà provvista anche di un'illustrazione del diagramma dei casi d'uso e di sequenza dell'applicativo web
- nella sezione numero 4.2 verrà trattata l'**implementazione dell'applicazione** con specifiche di come le varie librerie e framework, precedentemente illustrati nel Capitolo Background e Tecnologie usate, siano stati utilizzati all'interno dell'applicativo

4.1 Analisi e requisiti

Per compiere un'analisi esauriente è necessario valutare le vere e proprie funzionalità da implementare all'interno dell'applicazione, tenendo sempre a mente l'obiettivo principale precedentemente riportato.
Un **requisito** è una condizione, o capacità, necessaria a un utente per risolvere un problema. La condizione deve essere soddisfatta dal sistema per soddisfare un contratto.
I punti principali sono:

- il sistema deve fornire la possibilità all'utente di **scegliere un'immagine** dalla quale estrapolare del testo tramite OCR. Tale immagine può essere scelta tramite due modalità:
  - caricando una propria immagine sul server
  - scegliendo un'immagine online tramite URL
- il sistema deve fornire all'utente la possibilità di **scegliere la lingua** in cui è scritto il testo contenuto nell'immagine da far elaborare all'OCR. Tale scelta può essere effettuata tra una lista predefinita di lingue
- il sistema deve fornire la possibilità di **far riconoscere il testo** contenuto all'interno dell'immagine dall'OCR. Precondizione di questo requisito l'aver scelto un'immagine e una lingua
- il sistema deve fornire la possibilità di **modificare il testo** riconosciuto dall'OCR,
  - il testo riconosciuto può essere modificato nello stile
  - il testo riconosciuto può essere modificato nel contenuto

  Precondizione di questo requisito l'aver fatto riconoscere il testo all'OCR
- il sistema deve fornire la possibilità di **inserire metadati** riguardanti il testo riconosciuto dall'OCR riguardanti: informazioni come le condizioni fisiche del supporto del testo, le persone citate all'interno del testo e i luoghi citati all'interno del testo. Precondizione di questo requisito l'aver fatto riconoscere il testo all'OCR.
- il sistema deve fornire la possibilità di **scaricare documenti** nei tre formati sotto riportati. Precondizione di questo requisito l'aver fatto riconoscere il testo all'OCR.
  - **XML - TEI** –> contenente il testo riconosciuto dall'OCR corretto dall'utente, provvisto di stile e i metadati riguardanti il testo, inseriti dall'utente
  - **PDF** –> contenente il testo riconosciuto dall'OCR corretto dall'utente, non provvisto di stile e il titolo inserito con i metadati riguardanti il testo, inseriti dall'utente
  - **TXT** –> contenente il testo riconosciuto dall'OCR corretto dall'uten-

te, non provvisto di stile e nessuno dei metadati riguardanti il testo, inseriti dall'utente

- il sistema fornisce all'utente la possibilità di **copiare il testo** riconosciuto dall'OCR nella clipboard. Precondizione di questo requisito l'aver fatto riconoscere il testo all'OCR.

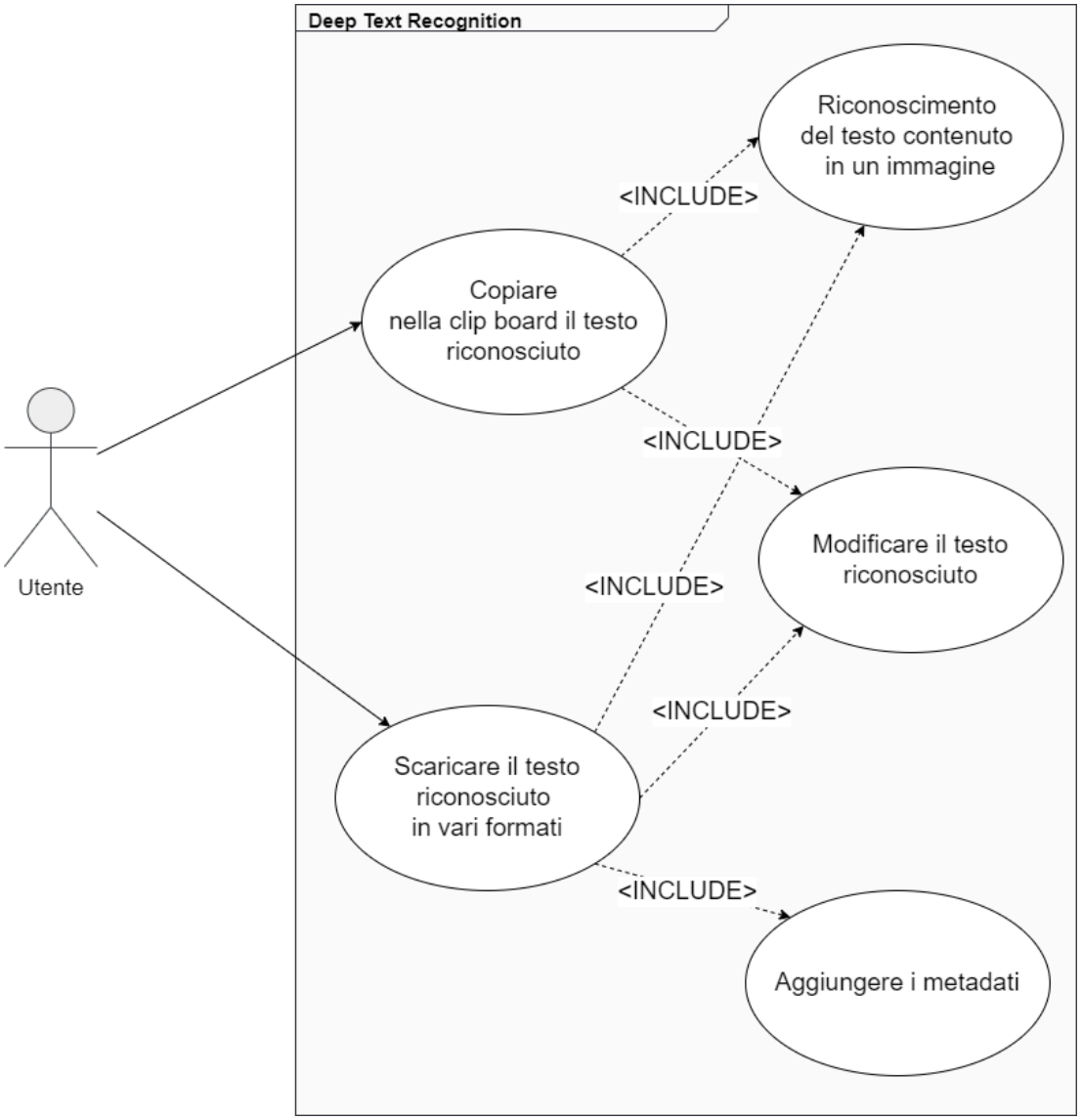


Figura 4.1: Il diagramma dei casi d'uso

Un modello è un'astrazione del sistema usato per specificare struttura e comportamento del sistema stesso. La modellazione dei requisiti funzionali dell'applicazione può essere rappresentata mediante un **diagramma dei casi d'uso** (figura 4.1, a pagina 79).
Un **diagramma dei casi d'uso** è dedicato a descrivere le funzioni e i servizi offerti dal sistema sulla base dei risultati osservabili, identifica cioè le variazioni di stato al verificarsi di alcune condizioni legate a due elementi l'attore e il caso d'uso.
L'**attore**, rappresentato nel diagramma dall'icona di un stickman (uomo stilizzato) rappresenta un'entità esterna, che interagisce con il sistema. L'unico attore all'interno del nostro sistema rappresenta l'utente, che vuole effettuare il riconoscimento di un documento.
I **casi d'uso**, rappresentati invece tramite degli ellissi, rappresentano una funzione o servizio, che deve essere garantito dal sistema, per soddisfare le necessità di uno o più attori.
All'interno del diagramma possiamo inoltre veder rappresentato come un rettangolo vuoto, il **sistema** stesso.

Un altro modo di rappresentare il funzionamento di un sistema è attraverso il **sequence diagram**, o diagramma di sequenza.
Il sequence diagram descrive le azioni che intercorrono fra attori, oggetti e entità del sistema che si sta rappresentando in termini di messaggi.
I messaggi vengono rappresentati all'interno del diagramma tramite delle frecce, rappresentate tratteggiate nel caso in cui si tratti di un messaggio di risposta.

Il diagramma di questo applicativo web, osservabile in figura 4.2 a pagina numero 81, contiene le interazioni fra l'utente e l'entità Deep Text Recognition, cioè l'applicazione web, dal momento successivo all'apertura del sito.

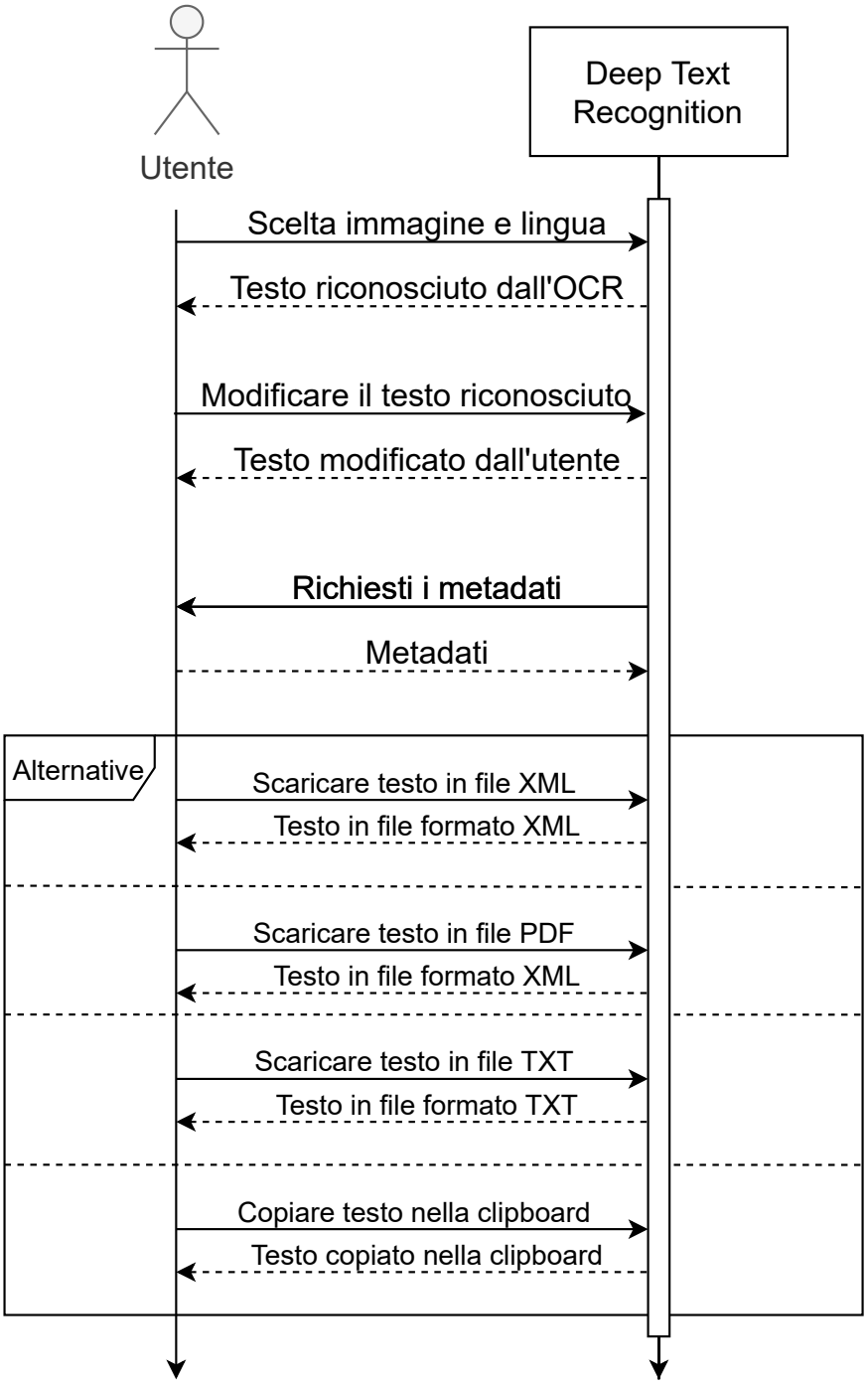


Figura 4.2: Il diagramma di sequenza

Vi sono inoltre i seguenti requisiti, che un utente deve soddisfare per poter utilizzare l'applicativo web:

- un dispositivo con cui poter navigare in rete
- una connessione internet
- conoscenze base riguardanti l'utilizzo di browser
- un'immagine contenente del testo da poter far riconoscere all'OCR, tale immagine può trovarsi in locale sul dispositivo dell'utente, oppure essere raggiungibile sul web tramite URL

4.2 Implementazione

All'interno di questa sezione viene illustrato come le tecnologie riportate nel capitolo Background e Tecnologie usate siano state utilizzate all'interno dell'applicativo. Questo capitolo sarà diviso in due macrosezioni: lato server, lato client.

4.2.1 Lato server

Lato server è stato scelto l'utilizzo del linguaggio, interpretato e general purpose, Python grazie alla sua forte flessibilità e al supporto delle sue molte librerie.

4.2.1.1 Flask

Per lo sviluppo lato server dell'applicazione web si è scelto l'utilizzo del microframework **Flask**, che permette di mappare URL (Uniform Resource Locator) specifici associandovi una funzione, all'interno della quale possono essere eseguite varie operazioni come la renderizzazione di un template grazie al motore Jinjia2.
Gli URL mappati dall'applicativo sono:

- **route '/'** –> per la visualizzazione della home page e la creazione e associazione del token contenente UML per il riconoscimento dell'utente, come spiegato di seguito
- **route '/upload'** –> route utilizzata per il caricamento dell'immagine sul server, può ritornare errore nel caso in cui a un utente non sia stato associato un codice univoco, o se ci sono errori nel caricamento del file. Tali errori vengono gestiti facendo visualizzare un errore all'utente tramite l'uso client side di SweetAlert

- **route '/users/img/<filename>'** –> utilizzato per la visualizzazione dell'immagine caricata dall'utente e salvata con il codice univoco associato a tale utente
- **route '/correction'** –> questa funzione ritorna renderizza il template "correction.html" passandogli come parametro il testo riconosciuto dall'OCR. Si tratta della pagina all'interno della quale l'utente ha la possibilità di modificare il testo riconosciuto dall'OCR. Nel caso in cui l'utente non possegga il codice univoco di riconoscimento viene sollevata un'eccezione e viene renderizzato un errore lato client tramite l'utilizzo di SweetAlert
- **route '/text'** –> questa route viene utilizza per la visualizzazione del template "create_xml.html", nuovamente nel caso in cui l'utente non possegga il codice univoco di riconoscimento viene sollevata un'eccezione e viene renderizzato un errore lato client tramite l'utilizzo di SweetAlert
- **route '/download'** –> a questa route è associata la funzione grazie alla quale l'utente può scaricare le varie versioni dei file, tale funzione utilizza il metodo statico "Text.create_file" illustrato di seguito. Anche in questa pagina nel caso in cui l'utente non possegga il codice univoco di riconoscimento viene sollevata un'eccezione e viene renderizzato un errore lato client tramite l'utilizzo di SweetAlert

4.2.1.2 PyJWT e modulo UUID

Grazie alla libreria **PyJWT**, che permette di codificare e decodificare JSON Web Tokens (JWT) usata in concomitanza con il **modulo Python UUID**, che fornisce oggetti UUID immutabili è stato sviluppato un sistema di riconoscimento degli utenti, che non viola la loro privacy e non li costringe a registrarsi all'interno del sito.
Viene salvato all'interno di un cookie criptato e non modificabile tramite HTTP il codice associato all'utente e tale codice viene poi associato ad ogni immagine caricata sul server da tale utente, in questo modo solamente il legittimo proprietario di un'immagine può visualizzarla (Code Listing 4.1).

```
1 def decode_token(req):
2     if 'token' in req.cookies:
3         return jwt.decode(req.cookies.get("token"),
4                           os.environ.get('JWT_SECRET'),
5                           algorithms=["HS256"])
6     else:
7         raise Exception
```

```
8
9  def encode_token():
10     return jwt.encode({'UUID': str(uuid.uuid4())},
11                       os.environ.get('JWT_SECRET'),
12                       algorithm=["HS256"])
```

Code Listing 4.1: Codifica e decodifica di token

All'interno di queste funzioni vengono usati i metodi:

- **jwt.encode("some": "payload", key, algorithm)** –> grazie a questa funzione il primo parametro viene codificato con l'algoritmo scelto come terzo parametro e la chiave scelta come seconda
- **jwt.decode(encoded, key, algorithms)** –> grazie a questa funzione il primo parametro viene decodificato con l'algoritmo scelto come terzo parametro e la chiave scelta come seconda, che devono entrambi essere gli stessi utilizzati per codificare il primo parametro
- **uuid.uuid4()** –> questa funzione genera un Universally Unique Identifier (identificativo universalmente univoco) casuale. Si è deciso di utilizzare la funzione uuid4() al posto della uuid1(), l'unica altra funzione del modulo uuid che genera un uuid casuale, perché quest'ultima può compromettere la privacy, in quanto crea un UUID contenente l'indirizzo di rete del computer

Alle funzioni vengono passati come parametri:

- **payload** *(il testo da codificare)* –> l'uuid generato da uuid.uuid4
- **chiave** –> come chiave per criptare e decriptare viene usata unavariabile d'ambiente chiamata JWT_SECRET e richiamata grazie al metodo "os.environ.get('JWT_SECRET')"
- **algoritmo** –> come algoritmo per criptare e decriptare si è scelto l'utilizzo di HS256 un algoritmo simmetrico, il che significa che invece di utilizzare una coppia di chiavi pubblica/segreta, utilizza un segreto condiviso per firmare e convalidare il token

4.2.1.3 PIL

Per la gestione delle immagini scelte dall'utente è stato scelto l'utilizzo di **PIL**, grazie alla sua compatibilità con l'OCR usato dall'applicativo, TesserOCR.

4.2.1.4 TesserOCR

TesserOCR è un wrapper Python dell'originale OCR **Tesseract** scritta per il C/C++, questo OCR è stato scelto grazie alla sua accuratezza superiore alla media e alla possibilità del riconoscimento di font e altri attributi dei caratteri, che offre.
Nel progetto questo OCR viene utilizzata all'interno della classe **Text.py**, strutturata come segue:

- gli attributi della classe, definite in Python tramite il decoratore @Property, che genera automaticamente per ogni proprietà le funzioni di get, set e delete, sono:
  - **file**: string–> il path o url all'immagine in cui il testo è contenuto
  - **lang**: string–> la lingua in cui il testo è scritto
  - **text**: string–> il testo riconosciuto dall'OCR all'interno del costruttore della Classe
  - **orientation**: int–> l'orientamento del testo in senso orario in gradi (0, 90, 180, 270) riconosciuto dall'OCR all'interno del costruttore della Classe
  - **direction**: int–> la direzione di scrittura del testo in gradi, riconosciuto dall'OCR all'interno del costruttore della Classe
  - **deskew_angle**: int–> l'angolo di raddrizzamento, dopo aver ruotato il blocco di testo in modo che sia verticale, di quanti radianti si deve ruotare il blocco in senso antiorario per essere livellato (-Pi / 4 <= deskew_angle <= Pi / 4). Anche questo attributo viene riconosciuto dall'OCR all'interno del costruttore
  - **font_per_word**: [(str, dict)] –> una lista composta da tuple composte dalle parole del testo una per volta e un dizionario contenente le informazioni sullo stile della parola (Code Listing 4.2)

```
1  lista [
2      ('parola',
3      {
4          'font_name': str,
5          'bold': bool,
6          'italic': bool,
7          'underlined': bool,
8          'monospace': bool,
9          'serif': bool,
10         'pointsize': int,
```

```
11          'font_id': int
12      })
13  ]
```

Code Listing 4.2: Struttura di font_per_word

- il **costruttore** della classe –> presi in input il path al file e la lingua riconosce tutti gli altri parametri illustrati precedentemente grazie all'OCR Tesseract, in particolare la libreria TesserOCR viene utilizzata nella sezione di codice 4.3

```
1  with PyTessBaseAPI(path='./tessdata-3.04.00/', lang=lang) as
       api:
2  image = Image.open(self.__file)
3  self.__text = tesserocr.image_to_text(image, lang=self.
       __lang)
4  api.SetImage(image)
5  api.Recognize()
6  self.__font_per_word = []
7  for word in iterate_level(api.GetIterator(), RIL.WORD):
8      symbol = word.GetUTF8Text(RIL.WORD)
9      word_attributes = word.WordFontAttributes()
10     if symbol and word_attributes is not None:
11         self.__font_per_word.append((symbol, word_attributes
               ))
12     else:
13         self.__font_per_word.append(None)
14 self.__orientation, self.__direction, self.__textline_order,
       self.__deskew_angle = api.AnalyseLayout().Orientation()
```

Code Listing 4.3: Costruttore della classe Text.py

All'interno del costruttore viene usato il modello tessdata-3.04.00, l'ultima utilizzabile per il riconoscimento degli attributi del font, grazie alla funzione di Tesseract "**WordFontAttributes**()".
Il testo viene riconosciuto dall'OCR grazie alla funzione "**Recognize**()".
Grazie ai metodi "**AnalyseLayout**().**Orientation**()" di Tesseract vengono inoltre riconosciuti direzione del testo, orientamento del testo, l'ordine delle righe di testo e l'angolo di raddrizzamento.

- **tagged_text** –> una funzione che ritorna il testo con i tag HTML relativi alle informazioni sui font delle singole parole determinati grazie alla funzione "WordFontAttributes()" dall'OCR nel costruttore della classe
- **other_format** –> un metodo statico che dato in input il testo con i tag HTML (ritornato dal metodo precedentemente illustrato tagged_text) e il formato desiderato (XML di default, altrimenti PDF, o TXT) ritorna il testo con i tag nel formato desiderato
- **create_file** –> un altro metodo statico che a partire dal text (precedentemente taggato grazie alle funzioni tagged_text e other_format), il formato in cui si desidera il file, i metadati relativi al testo e il nome del file, salva sul server un file in formato PDF, XML, o TXT contenete gli elementi precedentemente descritti.

4.2.1.5 PyPDF

Per la creazione dei file PDF creati dalla funzione create_file della classe Text viene utilizzata la libreria **PyFPDF**.
Di seguito la sezione di codice che si occupa della generazione dei pdf (Code Listing 4.4).

```
1 if format == 'PDF':
2     pdf = FPDF()
3     pdf.add_page()
4     pdf.add_font('Arial', '', 'Arial.ttf', uni=True)
5     pdf.set_font('Arial', size=15)
6     pdf.cell(200, 10, txt=data['title'], ln=1, align='L',
          markdown=True)
7     pdf.multi_cell(400, 10, txt=text, ln=1, align='L', markdown=
          True)
8     pdf.output(filename + '.pdf')
```

Code Listing 4.4: Creazione di pdf

All'interno del metodo statico "create_file" della classe "Text.py", vengono utilizzate le funzioni di questa libreria per creare un file pdf contenente il testo riconosciuto dall'OCR, corretto dall'utente e il titolo inserito fra i metadati.
Questa funzione serve a creare file contenenti testo e metadati di tipologie diverse a seconda della stringa formato, passata come parametro al metodo.

4.2.1.6 Modulo XML

Per quanto riguarda la creazione dei file XML viene utilizzata il **modulo XML** del Python standard all'interno della classe Xml, utilizzata per rappresentare un testo XML TEI, così strutturata:

- gli attributi della classe, definite in Python tramite il decoratore @Property, che genera automaticamente per ogni proprietà le funzioni di get, set e delete, sono:
  - **title**: str –> il titolo del testo
  - **publication_distributor**: str –> il distributore della pubblicazione del testo
  - **publication_availability**: str –> la disponibilità della pubblicazione del testo
  - **source_title**: list(string) –> i titoli delle fonti del testo
  - **source_publication_date**: list(string) –> le date di pubblicazione delle fonti del testo
  - **source_author**: list(string) –> gli autori delle fonti del testo
  - **text_description_channel**: string –> il canale con cui la il testo viene distribuito ('x': sconosciuto/inapplicabile, 'w': scritto, 's': parlato, 'sw': parlato per essere scritto, 'ws': scritto per essere parlato, 'm': mixed)
  - **text_description_constitution**: string –> il numero di parti di cui è costituito il testo ('single': una sola, 'composite': composto da più parti, 'frags', composto di frammenti, 'unknown': sconosciuto)
  - **text_description_derivation**: string –> derivazione del testo (": sconosciuto, 'translation': traduzione, 'abridgment': compendio, 'plagiarism': plagio, 'traditional': testo tradizionale)
  - **text_description_domain**: string –> argomento del testo ('art': arte, 'domestic': domestico o privato, 'religius': religioso o cerimoniale, 'business': business, 'education': educativo, 'govt': governativo o riguardante la legge, 'public': pubblico)
  - **text_description_purpose**: string –> lo scopo del testo (": unknown, 'persuade': persuasivo, 'express': espressivo 'inform': informativo, 'entartain': di intrattenimento)
  - **manuscript_settlement**: string –> la posizione del manoscritto
  - **manuscript_repository**: string –> il deposito in cui il manoscritto si trova
  - **manuscript_id**: string –> l'identificativo del manoscritto
  - **manuscript_content**: string –> il contenuto del manoscritto
  - **physical_object**: string –> il tipo di oggetto su cui il testo è scritto
  - **physical_material**: string –> il materiale di cui è fatto il supporto su

cui il testo è scritto
  - **physical_width**: string –> la larghezza del supporto su cui il testo è scritto
  - **physical_height**: string –> l'altezza del supporto su cui il testo è scritto
  - **physical_depth**: string –> la profondità del supporto su cui il testo è scritto
  - **physical_condition**: string –> le condizioni del supporto su cui il testo è scritto
  - **physical_layout**: string –> il layout con cui il testo è scritto
  - **physical_columns**: int –> il numero di colonne in cui il testo è scritto
  - **physical_ruled**: int –> il numero di righe disponibili sul supporto in cui il testo è scritto
  - **physical_written** (int) –> il numero di righe in cui il testo è scritto
  - **person_forename**: list(string) –> la lista dei nomi delle persone citate nel testo
  - **person_surname**: list(string) –> la lista dei cognomi delle persone citate nel testo
  - **person_role**: list(string) –> la lista dei ruoli delle persone citate nel testo
  - **person_birth**: list(string) –> la lista delle date di nascita delle persone citate nel testo
  - **person_death**: list(string) –> la lista delle date di morte delle persone citate nel testo
  - **person_sex**: list(string) –> la lista del sesso delle persone citate nel testo
  - **person_note**: list(string) –> la lista delle note sulle persone citate nel testo
  - **place_name**: list(string) –> la lista dei nomi dei luoghi citati nel testo
  - **place_name_not_before**: list(string) –> la lista delle date prima di cui il nome del luogo non è più valido
  - **place_name_not_after**: list(string) –> la lita delle date dopo cui il nome del luogo non è più valido
  - **place_country**: list(string) –> lo lista degli stati in cui si trovano i luoghi citati nel testo
  - **pace_region**: list(string) –> la lista delle regioni in cui si trovano i luoghi citati nel testo
  - **place_district**: list(string) –> la lista dei distretti in cui si trovano i luoghi citati nel testo
  - **place_note**: list(string) –> la lista delle note sui luoghi citati nel testo
  - **text**: string –> il testo
- il **costruttore** della classe –> presi in input il testo e tutti i metadati ri-

guardanti il testo all'interno di un dizionario appositamente composto
- **xml_to_file** –> preso in input il nome del file da creare, crea un documento in codifica XML TEI con tutte le informazioni contenute negli attributi dell'oggetto

4.2.2 Lato client

Lato client sono state sfruttati vari framework e librerie, di seguito elencati nelle varie sottosezioni.

4.2.2.1 Bootstrap

Per l'interfaccia dell'applicazione si è deciso di usare alcuni degli elementi del framework **Bootstrap**, che grazie alla loro facile personalizzazione risultano adatti alle esigenze e agli obiettivi del progetto.
Un altro aspetto fondamentale nella scelta di Bootstrap per l'interfaccia è stato il suo design responsive, che rende adatti i suoi elementi alla visualizzazione su piattaforme di qualsiasi dimensione e tipologia.

4.2.2.2 SweetAlert2

Come accennato nella sezione 4.2.1 per la visualizzazione di possibili errori, è stato deciso di utilizzare **SweetAlert2**, che grazie ai suoi alert completamente personalizzabili ed eleganti era l'alternativa migliore rispetto agli alert standard JavaScript, molto meno esplicativi e di più difficile comprensione.

Gli errori scatenati lato server da Python sono:

- **no cookie** –> nel caso in cui l'utente non abbia il codice univoco associato
- **no file** –> nel caso in cui ci sia un errore nel caricamento del file sul server

Tali errori vengono gestiti lato client con il codice 4.5

```
1 window.onload = (event) => {
2     //loading file error
3     if(ERROR == 'no_cookie')
4         swal("Session Error", "Please reload the page.", "error")
              ;
5     if (ERROR == 'no_file')
```

```
6         swal("File Error", "Please reload the page.", "error");
```

Code Listing 4.5: Errori gestiti lato client

Gli errori scatenati lato client sono invece:

- **no image** –> nel caso in cui l'utente provi a passare dalla prima alla seconda pagina prima di aver scelto un'immagine (Code Listing 4.6)

```
1 if(document.getElementById("chosen-img").src.includes("/img/
      example.png")) {
2     swal("Error", "No image has been chosen.", "error");
3 }
```

Code Listing 4.6: Errore No image

Il codice in questo snippet sta a significare che(?) se nella pagina di selezione dell'immagine è ancora visibile l'immagine di anteprima di default del sito, l'utente non ha ancora scelto un'immagine, quindi gli si impedisce l'accesso alla pagina successiva e si visualizza l'errore "Nessun'immagine è ancora stata scelta".

- **wrong url** –> utilizzato nel caso in cui nella pagina iniziale un utente inserisca un url non corretto, dalla funzione showImg, che si trova all'interno del file general.js (Code Listing 4.7)

```
1 if (url == 1 && !isImgLink(img)) {
2     swal("Wrong URL", "", "error");
3     return;
```

Code Listing 4.7: Errore wrong url

Il codice in questo snippet significa che(?) se img (il path dell'immagine nel server, o l'url dell'immagine) non è un link, ma viene comunicato dall'utente come URL valido, allora viene visualizzato l'errore "URL sbagliato".

4.2.2.3 NicEdit

Per dare la possibilità all'utente di modificare il testo riconosciuto dall'OCR è stato scelto l'utilizzo di **NicEdit**, una textArea che permette all'utente non solo

di modificarne il testo, ma anche lo stile, grazie al suo aspetto e utilizzo simile a un software di word processing come Word, o Pages.

Di seguito la sezione di testo con cui viene generato il NicEdit (Code Listing 4.8)

```
1 editor = nicEditors.editors.push(new nicEditor({buttonList: ['
    bold', 'italic', 'underline', 'ol', 'ul', 'fontSize', '
    forecolor', 'xhtml'], fullpanel : true}).panelInstance(
    document.getElementById('myNicEditor')));
```

Code Listing 4.8: NicEdit

Grazie a questo codice viene visualizzato un nicEditor con, sulla barra in alto, i pulsanti per (da sinistra verso destra):

- grassetto
- corsivo
- sottolineato
- liste
- liste numerate
- cambiare la dimensione del testo
- cambiare il colore del testo
- visualizzare la versione xhtml del testo

4.2.2.4 Perché è stato scelto di non utilizzare Vue.js

Alternativamente alla scelta di utilizzare Vue.js per la creazione di nuovi elementi dinamici è stato scelto di utilizzare le funzioni base del JavaScript per la creazione di elementi in modo sicuro, veloce e senza dover appesantire il sistema con una libreria complessa come Vue.js.

Capitolo 5

Guida all'uso dell'applicazione

All'interno di questo capitolo verrà accuratamente illustrato il metodo di utilizzo dell'applicativo web "Deep Text Recognition".
L'applicazione è stata realizzata seguendo una logica intuitiva, per permetterne un utilizzo facile ad ogni tipo di utente.

Per facilitare la comprensione della guida, il capitolo segue la stessa suddivisione delle pagine del sito, quindi in quattro macrosezioni:

- 5.1 **home** –> all'interno della pagina principale del sito è possibile scegliere l'immagine da far riconoscere all'OCR e la lingua del testo
- 5.2 **revisione del testo** –> all'interno della seconda pagina è possibile revisionare il testo riconosciuto dall'OCR per contenuto e stile
- 5.3 **metadati e download** –> all'interno della terza pagina dell'applicativo è possibile inserire i metadati riguardanti il testo e scaricare il documento in vari formati, oppure copiarlo nella clipboard, in modo da poterlo successivamente incollare
- 5.4 **about** –> all'interno della quale si possono trovare informazioni di carattere generale sul sito e la tesi

5.1 Home

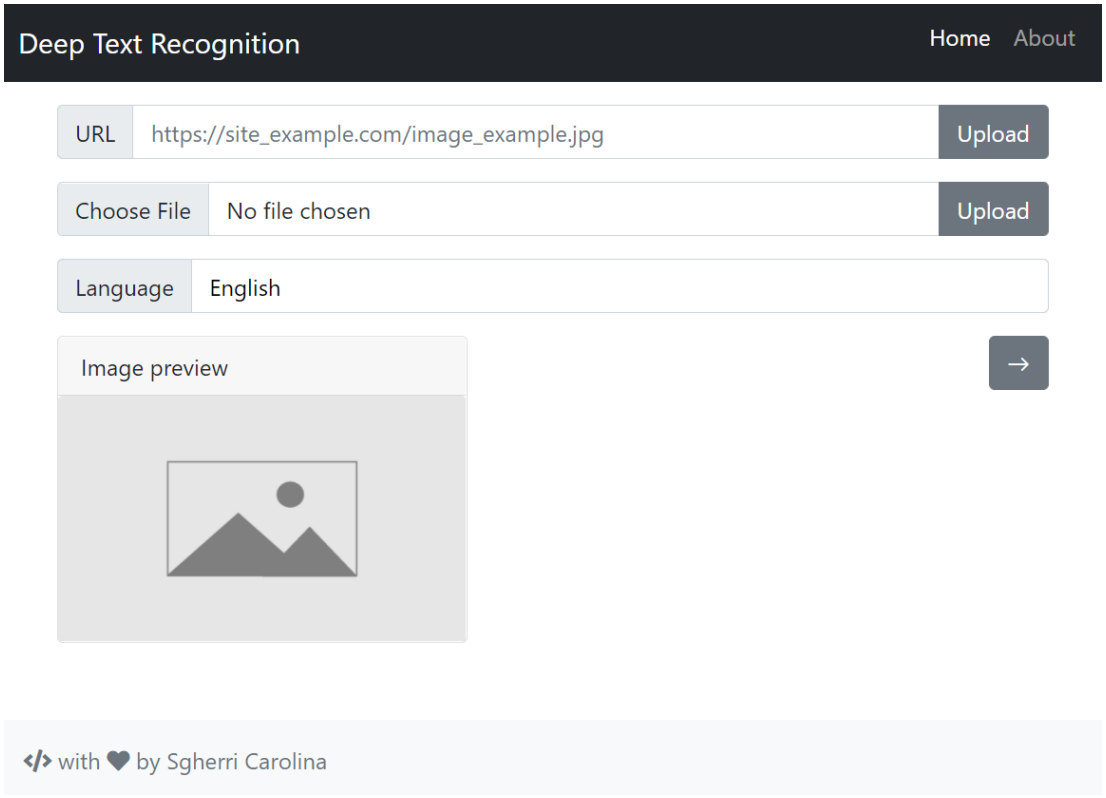


Figura 5.1: Pagina principale dell'applicativo web

All'interno della pagina principale del sito (figura 5.1 a pagina 94) possiamo notare due elementi che ci seguiranno all'interno di tutte le sezioni dell'applicativo: il **menu** di navigazione in alto attraverso il quale possiamo raggiungere la pagina Home e la pagina About e il **footer** in basso.

All'interno della pagina home possiamo **scegliere un'immagine** da far riconoscere all'OCR e la lingua del testo che l'immagine contiene. L'immagine può essere scelta sul web incollando poi il suo **URL** nella casella di input contrassegnata dall'etichetta URL e poi selezionata cliccando sul tasto Upload sulla destra.
Un altro modo per selezionare un'immagine è quello di scegliere di **caricarne una** presente sul proprio dispositivo, questo può essere fatto cliccando sull'apposita casella di input contrassegnata dall'etichetta Choose File, che permetterà all'utente di navigare fra i file del proprio file system; una volta scelta l'immagine si deve cliccare sul tasto upload alla destra della casella di input.
Una volta fatto l'upload dell'immagine potremo vedere una sua anteprima nel-

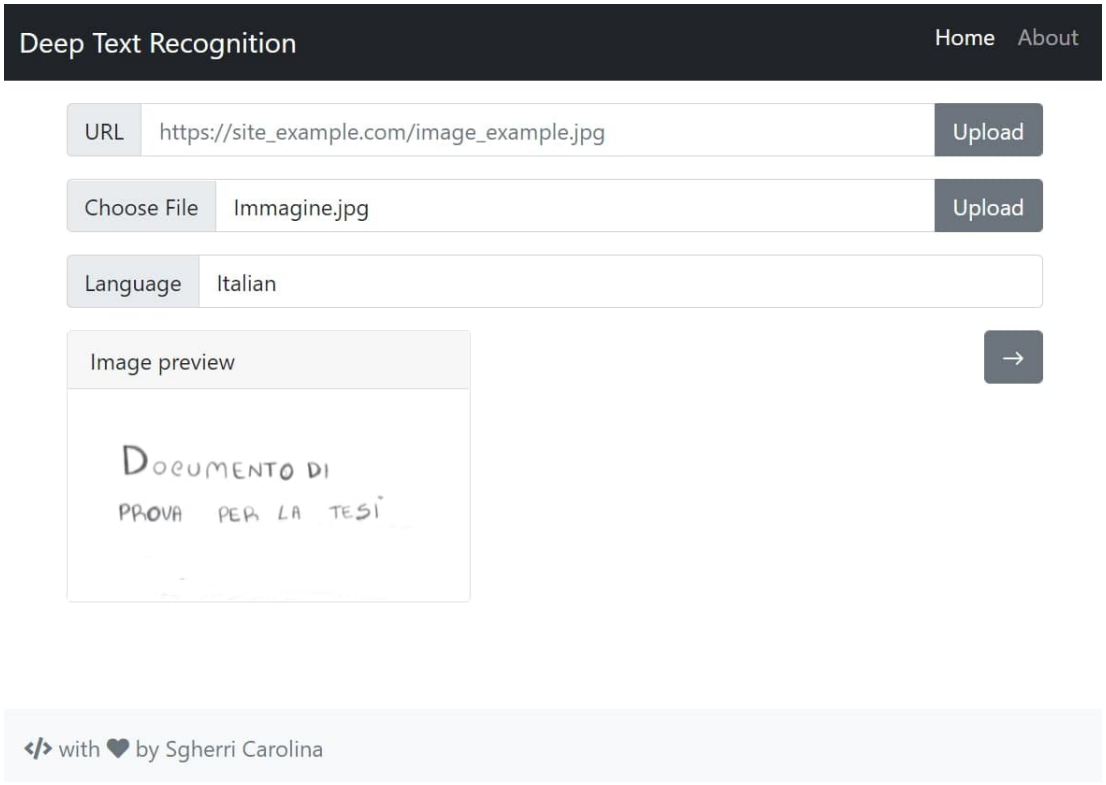


Figura 5.2: Selezione dell'immagine

la casella Image preview (figura 5.2, a pagina 95) comparire al posto dell'immagine segnaposto (il paesaggio grigio con le montagne).

Per **scegliere la lingua** deve essere usato l'apposito menu a tendina (figura 5.3 a pagina 96), che compare non appena si clicca sulla casella di input a destra dell'etichetta language.
Per velocizzare la selezione della lingua si può anche comporre il nome digitandolo sulla tastiera una volta cliccata l'apposita casella di input, ma deve comunque essere selezionata una delle lingue presenti, non ne possono essere aggiunte.
La lingua di default del sito è inglese.

Per **proseguire** nella navigazione del sito e far riconoscere all'OCR il testo nell'immagine scelta si deve cliccare sul tasto con la freccia verso destra, che si trova sotto la casella di input della lingua e a destra dell'anteprima dell'immagine.
Il caricamento della seconda pagina potrebbe richiedere qualche secondo a causa dell'elaborazione dell'OCR, si prega di non cliccare ripetutamente il tasto per proseguire alla pagina successiva, in quanto avrebbe solamente effetti

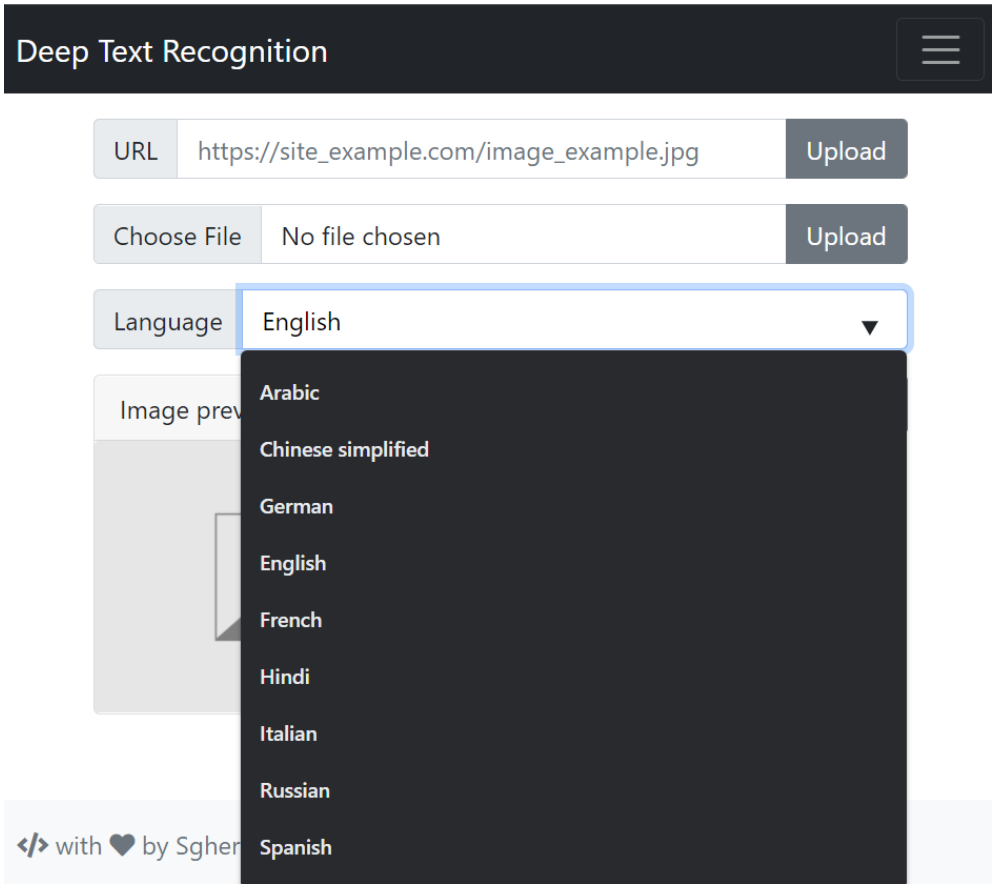


Figura 5.3: menu di selezione della lingua

negativi sulla durata del caricamento della pagina.

5.2 Revisione del testo

All'interno della seconda pagina del sito abbiamo il risultato del riconoscimento del testo da parte dell'OCR (figura 5.4, a pagina 97).
Questa pagina è divisa verticalmente in due parti, a sinistra si trova una **copia dell'immagine** scelta dall'utente da usare come riferimento e a destra si trova invece l'**editor** contente il risultato dell'OCR.

Il testo all'interno dell'editor è completamente personalizzabile da parte dell'utente, grazie agli strumenti nella barra in alto:

Figura 5.4: Pagina contenente i risultati dell'OCR

- **B** –> lo strumento Bold serve a rendere il testo in grassetto
- **I** –> lo strumento Italic serve a rendere il testo in corsivo
- **U** –> lo strumento Underline serve a rendere il testo in corsivo
- **bullet list** –> lo strumento Bullet list serve a creare una lista puntata
- **numbered list** –> lo strumento Numbered list serve a creare una lista numerata
- **font size** –> cliccando sullo strumento font size si apre una piccola finestra grazie alla quale è possibile cambiare la dimensione del testo (figura 5.5, a pagina 98)
- **Font color** –> grazie allo strumento font color, rappresentato da una tavolozza di colori, si può cambiare colore al testo (figura 5.6, a pagina 98)

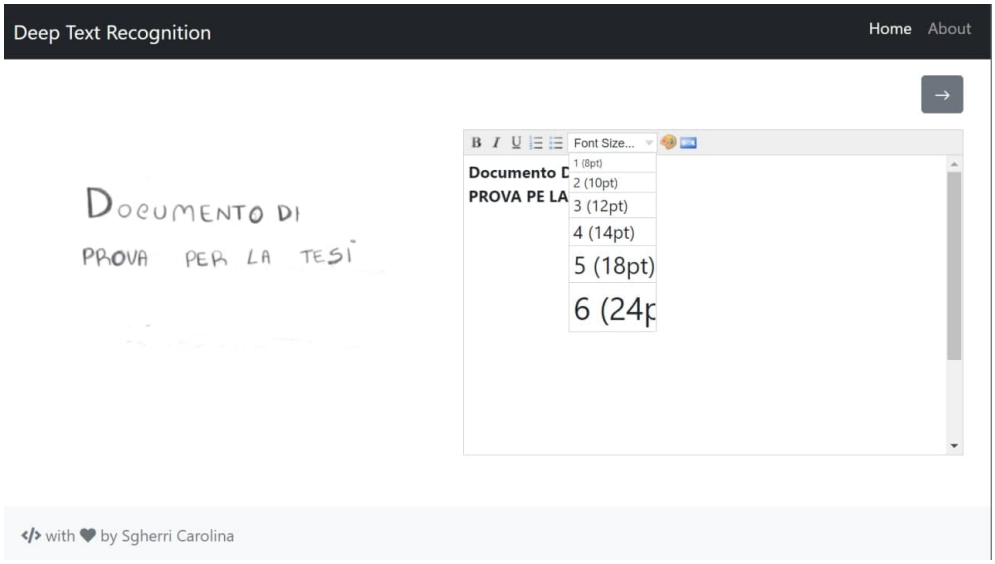


Figura 5.5: Modifica della dimensione dei caratteri

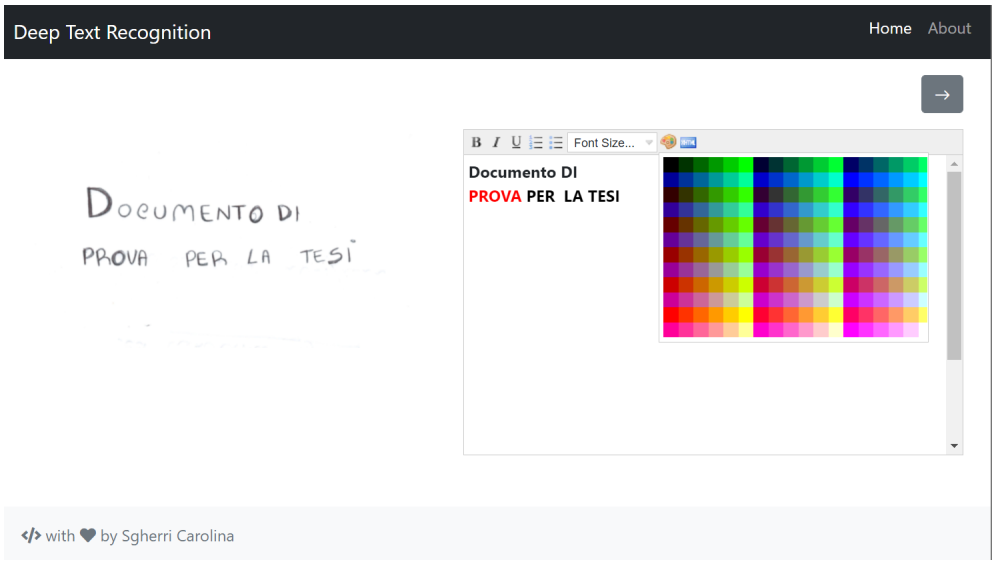


Figura 5.6: Modifica del colore del testo

Una volta completata l'operazione di correzione del testo e modifica dello stile, è possibile passare alla fase successiva cliccando sul bottone con la freccia verso destra, che si trova sotto il menu in alto e sopra l'editor di testo.

5.3 Metadati e Download

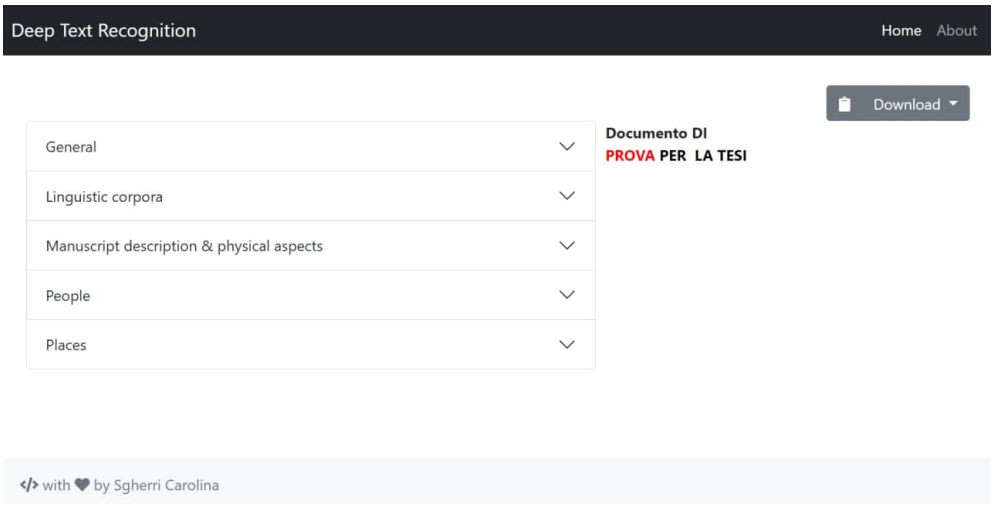


Figura 5.7: Pagina per l'inserimento dei metadati e download

Questa pagina è divisa verticalmente in due sezioni, a sinistra l'elemento accordion (fisarmonica) del quale usufruire per inserire i metadati e sulla destra il testo, riconosciuto dall'OCR e modificato dall'utente, e i bottoni per copiare il testo nella clipboard o per scaricarlo nei vari formati.

I **bottoni** visibili in figura 5.8 sono funzionanti come di seguito:

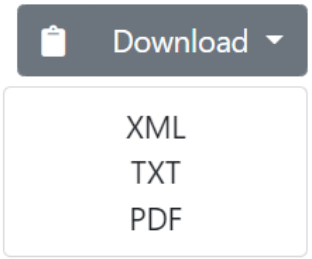


Figura 5.8: Bottoni per il download e per la copia in clipboard del testo

- **copiare** –> premendo sul tasto con il portablocchi si copia il testo nella clipboard e ciò lo rende incollabile dove più si desidera
- **scaricare** –> il documento può essere scaricato in vari formati, visualizzabili cliccando sul tasto "Download":
  - **XML** –> all'interno del quale vengono inseriti tutti i metadati compilati nell'elemento accordion (fisarmonica), che si trova a sinistra della pagina. Vengono inoltre inseriti tutti i tag relativi allo stile del testo
  - **PDF** –> all'interno del quale viene inserito il titolo, inserito nell'elemento accordion (fisarmonica), che si trova a sinistra della pagina

  - **TXT** –> all'interno del quale si trova solamente il testo privo di stile e metadati

L'elemento **accordion** (fisarmonica) è diviso in 5 diverse sezioni, apribili cliccando sopra il loro nome o sopra la freccia verso il basso alla destra del nome:

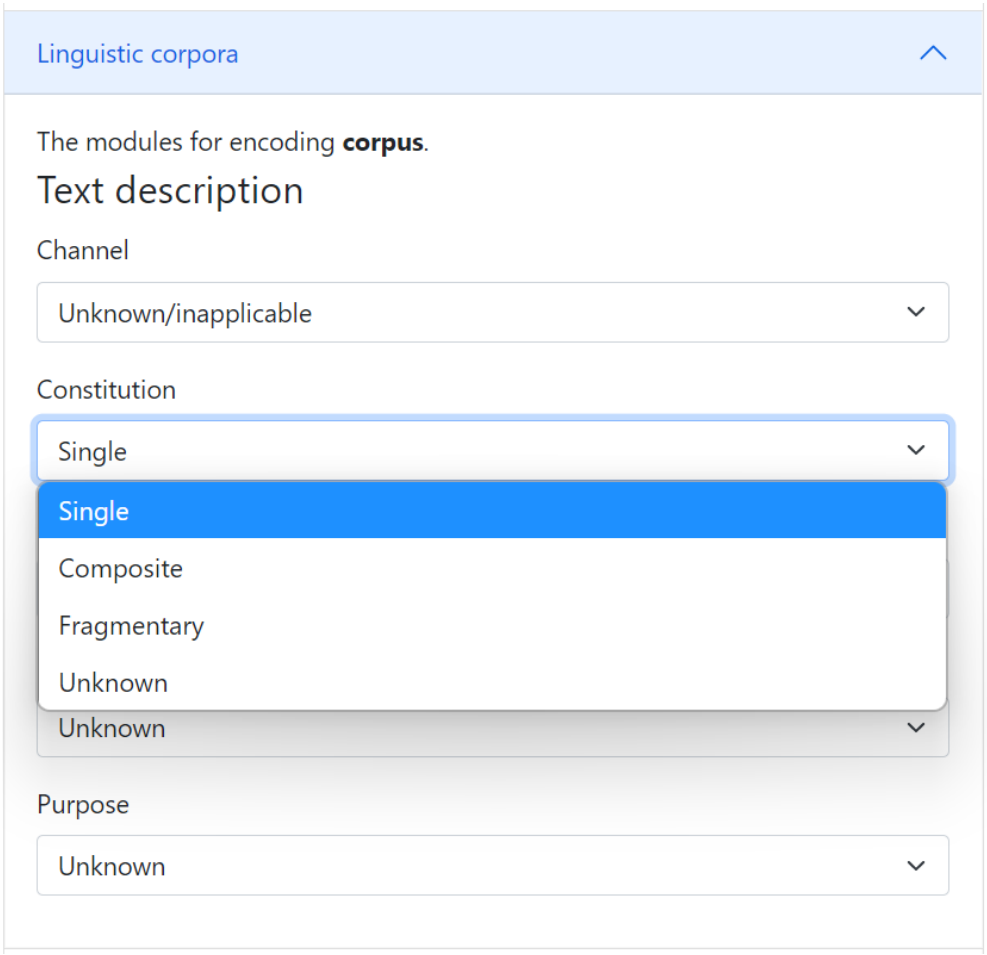


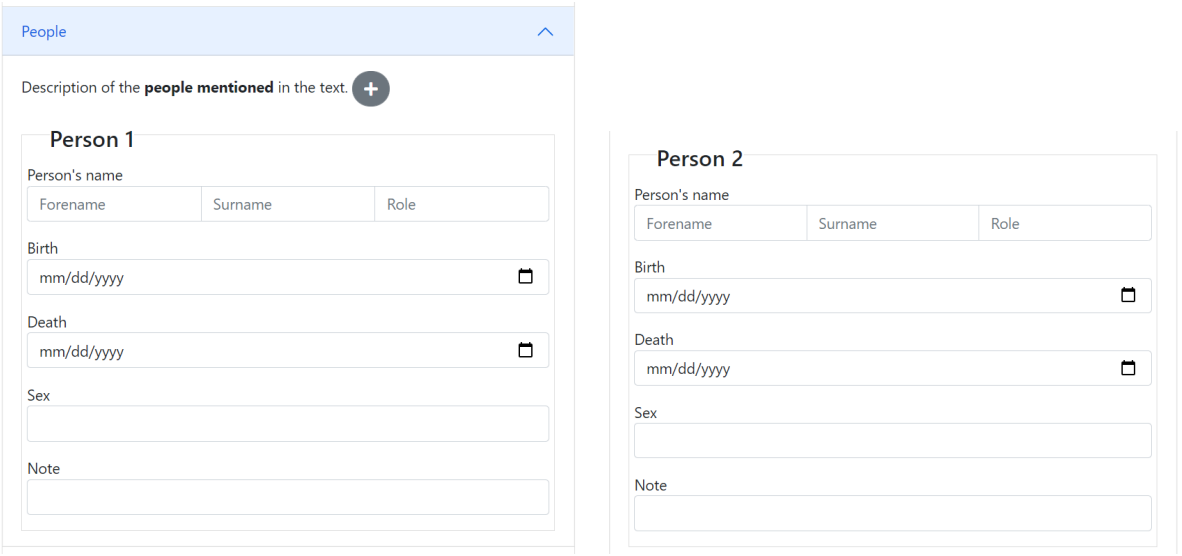
Figura 5.9: Sezione dedicata alla descrizione dei corpora linguistici

- **general** –> che permette la costruzione di uno schema molto scarno con il minimo assoluto di elementi. Gli elementi di questa sezione sono:
  - **title** –> il titolo del testo
  - **publication statement** –> dichiarazioni riguardanti la pubblicazione
    * **distributor** –> il distributore
    * **availability** –> la disponibilità

        * **date** –> la data
    - **sources description** –> descrizione delle fonti, è possibile aggiungere più fonti cliccando sul bottone con il +
        * **title** –> titolo della fonte
        * **author** –> autore della fonte
        * **publication date** –> data di pubblicazione della fonte
- **linguistic corpora** –> il modulo per la codifica di corpus[1] (figura 5.9 a pagina 100)
    - **channel** –> il canale di comunicazione del testo, fra le possibili scelte per questo campo c'è:
        * Unknown/inaplicable –> sconosciuto/inapplicabile
        * written –> scritto
        * spoken –> orale
        * spoken to be written –> orale per essere scritto
        * written to be spoken –> scritto per diventare orale
        * mixed –> misto
    - **constitution** –> il numero di parti in cui il testo è suddiviso, fra le possibili scelte per questo campo c'è:
        * single –> composto da un'unica parte
        * composite –> composto di più parti
        * fragmentary –> frammentario
        * unknown –> sconosciuto
    - **derivation** –> la derivazione del testo, fra le possibili scelte per questo campo c'è:
        * unknown –> sconosciuto
        * original –> originale
        * revision –> revisione
        * translation –> traduzione
        * abridgment –> riassunto
        * plagiarism –> plagio
        * traditional –> tradizionale
    - **domain** –> il dominio dei principali argomenti trattati nel testo, fra le possibili scelte per questo campo c'è:
        * unknown –> sconosciuto
        * art –> arte
        * domestic/private –> domestico/privato
        * religious/cerimonial –> religioso/cerimoniale
        * business –> affari
        * education –> educazione
        * government/law –> governo/legge

[1] una collezione di testi selezionati e organizzati per facilitare le analisi linguistiche

        * publilc –> pubblico
    - **purpose** –> lo scopo del testo, fra le possibili scelte per questo campo c'è:
        * unknown –> sconosciuto
        * persuade –> persuasivo
        * express –> espressivo
        * inform –> informativo
        * entartein –> intrattenitivo
- **manuscript description & physical aspects** –> questa sezione serve alla descrizione dei manoscritti e degli aspetti fisici complessi dei documenti
    - **manuscript description** –> descrizione del manoscritto
        * **manuscript identifier** –> l'identificativo del manoscritto, diviso in tre sottocampi: settlement(insediamento), repository(deposito), id (identificativo)
        * **manuscript content** –> una breve descrizione del contenuto del manoscritto
    - **physical description** –> descrizione fisica del documento
        * **object type** –> la tipologia di supporto su cui il testo è scritto
        * **manuscript material** –> il materiale di cui il supporto su cui il testo è scritto è fatto
        * **manuscript dimension** –> la dimensione del supporto su cui il testo è scritto, diviso in tre sottocampi: width(larghezza), height(altezza), depth(profondità)
        * **manuscript condition** –> una breve descrizione delle condizioni in cui il supporto su cui il testo è scritto
        * **layout description** –> una breve descrizione del layout del documento seguito da: columns (il numero delle colonne), ruled lines(il numero delle righe), written lines(il numero delle righe scritte)
- **people** –> questa sezione è dedicata alla descrizione dei personaggi presenti nel testo, un personaggio può essere aggiunto cliccando sul tasto +. Di ogni personaggio i campi da compilare sono:
    - **person's name** –> il nome della persona composto di forename (nome), surname (cognome), role (ruolo)
    - **birth** –> la data di nascita
    - **death** –> la data della morte
    - **sex** –> il sesso
    - **note** –> eventuali note sul personaggio



(a) I dettagli riguardanti la prima persona

(b) I dettagli riguardanti la seconda persona

Figura 5.10: I dettagli delle persone

- **places** –> questa sezione è dedicata alla descrizione dei luoghi presenti nel testo, un luogo può essere aggiunto cliccando sul tasto +. Di ogni luogo i campi da compilare sono:
  - **name** –> il nome seguito da not before (la data prima della quale il nome non è più valido), not after (la data dopo la quale il nome non è più valido)
  - **position** –> la posizione del luogo, è possibile specificare country (nazione), region (regione) e/o district (distretto)
  - **note** –> eventuali note sul luogo

5.4 About

La pagina About del sito è facilmente raggiungibile tramite il tasto "About", che si trova sulla destra nel menu in alto, presente in tutte le pagine.
All'interno di questa pagina si trovano delle informazioni di carattere generale riguardanti il sito stesso e il motivo per cui è stato creato.

Capitolo 6

Conclusioni

Il fenomeno sempre crescente dell'uso di tecnologie innovative nel campo delle Digital Humanities, ha portato a una riduzione di tempi e costi nelle ricerche e nelle elaborazioni umanistiche, ciò ha fatto sì, che il patrimonio artistico venisse esaltato e reso maggiormente accessibile, garantendo così una più approfondita conoscenza delle tematiche culturali a un maggior numero di persone. É quindi di fondamentale importanza il sostegno sia da parte della comunità scientifica, che da parte di quella umanistica a quel campo di ricerca che sono le Digital Humanities, che sta dando possibilità di sviluppi incommensurabili al patrimonio artistico e culturale del mondo.

Durante la realizzazione dell'applicativo è stata riscontrata la problematica della qualità degli OCR (Optical Character Recognition - riconoscimento ottico dei caratteri), illustrata anche nella Sezione 2.1.
La qualità del testo convertito dagli OCR è una componente critica per la conservazione del patrimonio storico culturale. Una qualità insoddisfacente significa testo non ricercabile, o analizzabile correttamente.

L'unica soluzione ad oggi attuabile è la post correzione, adottata anche all'interno del sito, per mezzo della seconda pagina dell'applicativo, all'interno della quale è l'utente a poter realizzare questa correzione tramite l'editor.
La post correzione di archivi storici viene invece solitamente realizzata utilizzando modelli di deep learning con elevata precisione, tecnica però applicabile solamente nel caso in cui sia possibile ottenere un set di dati appropriato.

6.1 Note di sviluppi futuri

L'applicativo realizzato altro non è che una prima implementazione, una solida base, per infiniti nuovi utilizzi e nuove feature.

L'obiettivo di questa tesi è quello di dimostrare l'importanza della collaborazione fra i mondi dell'informatica e delle digital humanities per l'esaltazione del patrimonio culturale e il raggiungimento di un numero sempre maggiore di individui, con la divulgazione realizzata grazie all'uso delle nuove tecnologie.

Lavori futuri sull'implementazione potrebbero essere:

- l'aggiunta di possibilità di far registrare gli utenti così da permettergli di riprendere la modifica di testi, o di poter scaricare nuovamente immagini precedentemente convertite
- l'aggiunta della possibilità di caricare i testi convertiti direttamente all'interno del cloud utilizzato dall'utente (Google drive, Cloud, ...)
- la possibilità di scaricare il testo in più formati come per esempio doc (microsoft word) e odt (open document text - documento di testo aperto)
- ottimizzare il riconoscimento del testo da parte dell'OCR tramite metodi di elaborazione delle immagini per migliorarne la qualità, come la trasformazione in bianco e nero
- aggiunta di controllo ortografico e correzione automatica degli errori più comuni dell'OCR
- l'aggiunta di una nuova sezione del sito all'interno della quale si dà la possibilità all'utente di creare un file XSLT[1](eXtensible Stylesheet Language - Linguaggio estensibile per fogli di stile) personalizzabile tramite un'interfaccia grafica, grazie al quale creare una grafica per il documento xml

6.1.1 I vantaggi del quantum computing

Sebbene l'utilizzo dei computer classici nelle discipline umanistiche digitali sia stato applicato con successo per affrontare alcuni problemi, le prestazioni di questi computer non sono paragonabili al potenziale dei computer quantistici.

[1] un linguaggio di programmazione che rende possibile la trasformazione di un documento XML in un altro documento, per esempio HTML, aggiungendovi uno stile

Il quantum computing, nato nel 1980, (computazione quantistica) è un tipo di calcolo che sfrutta le **proprietà degli stati quantistici**, come sovrapposizione di stati, interferenza e entanglement.
L'uso dei computer quantistici offre un grande potenziale perché:

- sono molto **più veloci** dei computer classici nella risoluzione di alcuni problemi complessi
- le soluzioni possono essere molto **più precise**
- consentono la soluzione di classi di problemi che possono difficilmente essere risolti dai computer classici e di **problemi** che possono essere **risolti solamente su un computer quantistico**
- il loro utilizzo promette di essere molto **più economico** di quello **dei super-computer** [2, 40]

Poiché si prevede che i computer quantistici diventeranno disponibili generalmente su larga scala negli anni a venire, è promettente l'idea di utilizzare i vantaggi di questa nuova tecnologia per affrontare questioni esistenti e completamente nuove nelle discipline umanistiche. [2]

Per quanto riguardalle applicazioni nella ricerca, sono diversi i domini in cui i computer quantistici sono già utilizzati come ad esempio nell'**artificial intelligence** e nel **deep learning**.
Un computer quantistico può già ad oggi **accelerare notevolmente** l'apprendimento automatico non supervisionato, in particolare per il clustering[2]. [2]

[2]selezione e raggruppamento di elementi omogenei in un insieme di dati

Bibliografia

[1] Roberto Busa e associati. *Index Thomisticus*. URL: https://www.corpusthomisticum.org/it/index.age.

[2] Johanna Barzen e Frank Leymann. *Quantum humanities: a vision for quantum computing in digital humanities*. Ago. 2019. URL: %5Curl%7Bhttps://link.springer.com/article/10.1007/s00450-019-00419-4%7D.

[3] Francois Chollet. *Deep learning with Python*. 2017.

[4] Alex Clark e Contributors. *Pillow (PIL Fork) 8.4.0 documentation*. URL: https://pillow.readthedocs.io/en/stable/.

[5] TEI Consortium. *The TEI Guidelines*. URL: https://sweetalert2.github.io/.

[6] Flip111 Fayez Zdenop Simonflueckiger Johnthagen e Aggie Bill. *TesserOCR - GitHub*. URL: https://github.com/sirfz/tesserocr/blob/master/README.rst.

[7] *Flask Documentation (2.0.x)*. URL: https://flask.palletsprojects.com/en/2.0.x/.

[8] *FPDF for Python*. URL: https://pyfpdf.readthedocs.io/en/latest/.

[9] *FPDF library - PDF generator*. URL: http://www.fpdf.org/.

[10] Jr. Fred L. Drake e Python community. *Python 3 documentation*. URL: https://docs.python.org/3/.

[11] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[12] Chazzz Harald Scheidl Nishant Bhandari e Reed Jones. *Simple HTR - GitHub*. URL: https://github.com/githubharald/SimpleHTR#readme.

[13] Vivian Hu. *Bootstrap 5 Removes jQuery Dependency*. A cura di InfoQ. Ago. 2020. URL: %5Curl%7Bhttps://www.infoq.com/news/2020/08/bootsrap-5-drops-jquery/%7D.

[14] *Keras OCR - Tensor Flow Hub*. URL: https://tfhub.dev/tulasiram58827/lite-model/keras-ocr/dr/2.

[15] Jacob Thornton Mark Otto e Bootstrap contributors. *Introduction · Bootstrap v5.0*. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/.

[16] Mozilla. *CSS Documentation*. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

[17] Mozilla. *HTML Documentation*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[18] Mozilla. *JavaScript Documentation*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[19] Andreas C Müller e Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[20] *NicEdit Inline WYSIWYG Documentation*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[21] Latex Ninja. *Machine Learning for the Humanities: A very short introduction and a not-so-short reflection*. 2020. URL: %5Curl%7Bhttps://latex-ninja.com/2020/10/25/machine-learning-for-the-humanities-a-very-short-introduction-and-a-not-so-short-reflection/%7D.

[22] Maayan Zhitomirsky-Geffet Omri Suissa Avshalom Elmalech. *Text analysis using deep neural networks in digital humanities and information science*. Giu. 2021. URL: %5Curl%7Bhttps://asistdl.onlinelibrary.wiley.com/doi/full/10.1002/asi.24544?saml_referrer%7D.

[23] Python. *PEP 20 – The Zen of Python*. URL: https://www.python.org/dev/peps/pep-0020/.

[24] Pool David; Mackworth Alan; Goebel Randy. *Computational Intelligence: A Logical Approach*. A cura di Oxford University Press. 1998. URL: %5Curl%7Bhttps://www.cs.ubc.ca/~poole/ci/ch1.pdf%7D.

[25] Google Research. *Ray Smith*. URL: https://research.google/people/author4479/.

[26] *Rosetta - Tensor Flow Hub*. URL: https://tfhub.dev/tulasiram58827/lite-model/rosetta/dr/1.

[27] *SweetAlert2 - a beautiful, responsive, customizable and accessible (WAI-ARIA) replacement for JavaScript's popup boxes*. URL: https://sweetalert2.github.io/.

[28] Tesseract-ocr. *Tesseract User Manual | tessdoc*. URL: https://tesseract-ocr.github.io/tessdoc/.

[29] *Welcome to PyJWT*. URL: https://pyjwt.readthedocs.io/en/latest/.

[30] Wikipedia. *Artificial neural network*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Artificial_neural_network%7D.

[31] Wikipedia. *Bootstrap (framework)*. URL: %5Curl%7Bhttps://it.wikipedia.org/wiki/Bootstrap_(framework))%7D.

[32] Wikipedia. *Deep Learning*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Deep_learning%7D.

[33] Wikipedia. *Digital Humanities*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Digital_humanities%7D.

[34] Wikipedia. *Flask (web framework)*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Flask_(web_framework)%7D.

[35] Wikipedia. *Javascript*. URL: %5Curl%7Bhttps://it.wikipedia.org/wiki/JavaScript%7D.

[36] Wikipedia. *Machine Learning*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Machine_learning%7D.

[37] Wikipedia. *OCR*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Optical_character_recognition#Pre-processing%7D.

[38] Wikipedia. *Pillow Imaging Library*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Python_Imaging_Library%7D.

[39] Wikipedia. *Python (programming language)*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Python_(programming_language)%7D.

[40] Wikipedia. *Quantum Computing*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Quantum_computing%7D.

[41] Wikipedia. *Text Encoding Initiative*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Text_Encoding_Initiative%7D.

[42] Wikipedia. *Vue.js*. URL: %5Curl%7Bhttps://en.wikipedia.org/wiki/Vue.js%7D.

[43] Evan You e VUE Team. *Introduction | Vue.js*. URL: https://v3.vuejs.org/guide/introduction.html#what-is-vue-js.