Digitalizzazione di un fondo archivistico per la creazione di un centro di documentazione digitale

Dario Baldini
Università degli Studi di Roma Tor Vergata/Fondazione Giannino Bassetti, Italia - baldini.dario98@gmail.com

ABSTRACT (ITALIANO)

Questo progetto di dottorato è incentrato sulla digitalizzazione di un fondo archivistico per creare un centro di documentazione consultabile con metodi informatici, legati principalmente a tecnologie web e di intelligenza artificiale. Innanzitutto, le carte sono state scansionate e i file di immagine sono elaborati da *Image* API dello standard IIIF per l'invio delle immagini e delle informazioni tecniche in JSON; esse sono state programmate con una applicazione Python scritta in Flask. Le scansioni sono state strutturate, descritte e annotate tramite *Presentation* API, file JSON che poi vengono elaborati dal *viewer* Mirador 3 per generare un'interfaccia web per la fruizione dei documenti digitalizzati; viene inizializzato e configurato dinamicamente in un *template* Jinja2. Dopo di che, il testo dei documenti è stato trascritto con sistemi OCR e codificato nello schema XML/TEI in modo molto granulare, marcando specifiche porzioni testuali per registrarne il valore semantico. È stato poi programmato un complesso *style sheet* XSL per la trasformazione dei file XML in XHTML, in modo che i documenti codificati possano essere visualizzati sul web, che il loro aspetto grafico possa essere definito con *style sheet* CSS e che possano essere arricchiti di funzionalità interattive con JavaScript. Successivamente, la ricerca si concentrerà sulle possibilità di applicazione dei sistemi di AI ai prodotti digitali ottenuti durante le fasi precedenti, utilizzando LLM *open-source* in locale grazie a specifiche piattaforme. Verranno implementate funzionalità interattive al centro di documentazione per elaborazioni testuali come traduzione e sintesi. Verrà programmata una interfaccia per il recupero delle informazioni basata su sistemi di RAG per estrarre i dati semantici della codifica testuale. Si valuterà la fattibilità del *fine-tuning* per permettere a un modello di riconoscere le componenti testuali, anche per l'automatizzazione delle operazioni di codifica.
Parole chiave: Archivio digitale; *International Image Interoperability Framework* (IIIF); *Text Encoding Initiative* (TEI); sviluppo web dinamico; intelligenza artificiale

ABSTRACT (ENGLISH)

Digitisation of an archive collection for the creation of a digital documentation centre

This PhD project focuses on the digitisation of an archive collection to create a documentation centre that can be consulted with IT methods, mainly related to web and artificial intelligence technologies. First, the papers were scanned, and the image files are processed by Image APIs of the IIIF standard to send images and technical information in JSON; they have been programmed using a Python application written in Flask. The scans have been structured, described and annotated via Presentation APIs, JSON files that are then processed by the Mirador 3 viewer to generate a web interface for the use of the digitised documents; it is initialized and configured dynamically in a Jinja2 template. After that, the text of the documents has been transcribed with OCR systems and encoded meticulously in the XML/TEI schema, marking specific textual portions to record their semantic value. A complex XSL style sheet has then been programmed to transform the XML files into XHTML, so that the encoded documents can be displayed on the web, their graphic appearance can be defined with CSS style sheets and they can be enriched with interactive features with JavaScript. Subsequently, the research will focus on the possibilities of applying AI systems to the digital products obtained during the previous phases, using open-source LLM locally thanks to specific platforms. Interactive features will be implemented at the documentation centre for text processing such as translation and synthesis. An information retrieval interface based on RAG systems will be programmed to extract semantic data from text encoding. The feasibility of fine-tuning will be evaluated to allow a model to recognize textual components, also to automate the encoding process.
Keywords: digital archive; International Image Interoperability Framework (IIIF); Text Encoding Initiative (TEI); dynamic web development; artificial intelligence

L'obiettivo di questo progetto di dottorato è la digitalizzazione di un fondo archivistico per la creazione di un centro di documentazione consultabile con metodi informatici, legati principalmente a tecnologie web e di intelligenza artificiale. Il processo di digitalizzazione consiste nell'applicazione di due standard

internazionali molto noti e utilizzati nel campo delle *Digital Humanities*, IIIF e TEI, e di una serie di sistemi di sviluppo web dinamico.

1. Digitalizzazione dei documenti

Il progetto può essere suddiviso in tre fasi ben distinte: la prima riguarda la scansione dei documenti con uno scanner professionale e la loro codifica negli standard IIIF e TEI, impiegati proprio per la conservazione e la trasmissione del patrimonio culturale. L'*International Image Interoperability Framework* (IIIF) è un insieme di API sviluppate per la trasmissione di immagini e contenuti multimediali di vario tipo ad ambienti web diversi, garantendo la massima compatibilità (Meschini, 2020, "Una domanda ancora aperta"). Dopo aver acquisito le immagini, i file vengono innanzitutto caricati su server e organizzati in *folder*, e per permetter la loro elaborazione sono stati programmati *Image* API IIIF grazie a un'applicazione Python; quest'ultima utilizza il *micro-framework* Flask, che attraverso il sistema del routing, ovvero mappando specifici URL alle funzioni (Pallets, 2024, "Routing"), permette lo scambio dei dati di tutte le componenti del progetto tramite richieste HTTP. In particolare, sono state programmate due funzioni per i due *web service* definiti dalle linee guida dello *Image* API: lo *Image Request*, per richiedere una rappresentazione dell'immagine, e lo *Image Information Request*, per richiedere informazioni tecniche sull'immagine in formato JSON-LD. Per il primo, è stato mappato uno URI conforme alla sintassi dello *Image Request* a una funzione per elaborare l'immagine. I parametri di prefisso — ossia il percorso — e identificatore vengono estratti dalla richiesta HTTP e utilizzati nella funzione per selezionare l'immagine, mentre gli altri parametri di regione, dimensioni, rotazione e qualità specificano quali operazioni di trasformazione eseguire: la porzione da rendere; come ingrandire o rimpicciolire l'immagine; come ruotarla o specchiarla; come modificarne i colori. L'immagine manipolata viene infine inviata in formato JPEG. Per il secondo, è stato mappato uno URI conforme alla sintassi dello *Image Information Request* a una funzione con un dizionario Python con delle proprietà preconfigurate, ovvero `@context` che fa riferimento allo URI del contesto dello Image API, la versione dell'*Image* API e il livello di *compliance*, l'insieme di possibilità di trasformazione che vengono garantite. Prefisso e identificatore vengono estratti dalla richiesta HTTP e inseriti nel dizionario, insieme alle dimensioni che vengono calcolate analizzando l'immagine. Tutte le informazioni tecniche vengono poi inviate in formato JSON (Appleby et al, 2020a). Dopo di che, le immagini vengono strutturate e descritte attraverso il *Presentation* API, anch'esso da programmare nel formato JSON-LD. La struttura tipica di un *Presentation* API è composta da un *manifest*, che codifica un intero oggetto composito, che contiene in rapporto di *embedding* una serie di *canvas*, dei contenitori digitali che offrono rappresentazioni di parti specifiche dell'oggetto composito. Nel progetto, lettere, note e documentazione varia vengono raccolti insieme, mentre testi più importanti hanno file dedicati. La prima proprietà di un *Presentation* API deve essere `@context`, che rimanda al contesto del *Presentation* API; le uniche proprietà obbligatorie in tutte le risorse sono `id` e `type`. È anche possibile programmare delle collezioni, risorse che raccolgono *manifest* e collezioni subordinate con un rapporto di *referencing*, cioè facendo riferimento alla loro proprietà `id`. All'interno della proprietà `items` delle *canvas* si trovano le pagine di annotazioni, la cui unica funzione è contenere le annotazioni, sistemi di codifica in JSON-LD definiti dal *Web Annotation Data Model* per legare informazioni aggiuntive a risorse online, o per descrivere le associazioni esistenti tra loro (Sanderson et al, 2017); esse sono composte da contenuti audiovisivi o testuali di vario tipo, il *body*, legati a una *canvas*, il *target*, per permetterne la visualizzazione. Per legare le singole scansioni alle *canvas* è necessario utilizzare il valore `painting` nella la proprietà `motivation` dell'annotazione, riportare le informazioni tecniche dell'immagine — tra cui URI di *Image* API — nel *body*, e riprendere la proprietà `id` delle *canvas* stesse come *target*; la proprietà service verrà usata all'interno del *body* per fare riferimento al servizio dell'*Image* API. È anche possibile annotare specifiche parti delle scansioni inserendo le pagine di annotazioni nella proprietà `annotations` delle canvas, utilizzando il valore `painting` nella la proprietà `motivation` dell'annotazione, scrivendo un contenuto testuale nel *body*, e selezionando una porzione di *canvas* come *target*, riprendendo la proprietà `id` seguita da un *FragmentSelector*. Questa tecnica è stata utilizzata per evidenziare e descrivere simboli, firme o altri elementi grafici dei documenti. Infine, le *canvas* possono essere elencate all'interno di un *manifest* in modi alternativi con i *range*. Nel progetto sono stati sfruttati per ricreare digitalmente i sommari dei documenti originali, o per raccogliere i documenti a seconda della tipologia (corrispondenza, note, etc.) (Appleby et al, 2020b).

Successivamente si procede alla trascrizione dei documenti con sistemi OCR e alla codifica testuale con il linguaggio di marcatura definito dalla *Text Encoding Initiative* (TEI), la comunità internazionale che si occupa della gestione, dello sviluppo e della promozione delle TEI *Guidelines*, un insieme standardizzato di

pratiche di codifica per materiale testuale pensato per la produzione di documenti archiviabili e trasmissibili con metodi informatici, soprattutto nell'ottica della conservazione a lungo termine di testi antichi (Sechi, 2010, "Text Encoding Initiative"). Un documento TEI riprende la struttura base e le regole di sintassi di XML: si inserisce la dichiarazione XML alla prima linea, seguita da un solo elemento radice (Sechi, 2010, "Struttura di XML"), marcato con il *tag* `<TEI>`, che racchiude un solo TEI *header*, in cui vengono registrati i metadati, e uno o più elementi in cui riportare il testo dei documenti (TEI Consortium, 2025, "Default Text Structure"). Come per i *manifest*, lettere, note e documentazione varia vengono raccolti in un unico file TEI, mentre testi più importanti hanno file dedicati. Il TEI *header* si compone di cinque parti: *file description*, in cui si descrivono il documento TEI e le fonti e che contiene gli unici elementi veramente obbligatori, ovvero `<titleStmt>`, `<publicationStmt>` e `<sourceDesc>`; *encoding description*, in cui si riassumono pratiche editoriali e metodologie di codifica; *text profile*, in cui si riportano informazioni classificatorie e contestuali; una parte marcata dal *tag* `<xenoData>` in cui inserire metadati in un formato non conforme alle specifiche TEI; *revision history*, in cui si elencano le revisioni del documento (*Ivi*, "The TEI Header"). La marcatura del testo può essere considerata un processo di modellizzazione per come viene concepito nelle *Digital Humanities*, pensato quindi per l'elaborazione elettronica e la rappresentazione dei testi (Ciula et al., 2023, "Modelling Practices"). La segmentazione nelle unità testuali di livello più generale prevede l'utilizzo dell'elemento *division* `<div>`, mentre le vere e proprie porzioni di testo vengono segmentate con elementi di varie categorie: *chunk*, *phrase-level*, *inter-level* e *component*. Tutti i vari fenomeni testuali che devono essere differenziati dal testo che li circonda saranno codificati con i relativi elementi *inline* (TEI Consortium, 2025, "The TEI Infrastructure"): nomi propri di persona, organizzazione e luogo; numeri, misure e date; termini stranieri e tecnici. Allo stesso modo, verranno impiegati anche tutti quegli elementi che rappresentano interventi editoriali, legati al trattamento di abbreviazioni, forme erronee, aggiunte e cancellazioni, ma anche interpretativi per casi in cui il testo risulta illeggibile. In tutti i casi si mira a una codifica abbastanza minuziosa dei fenomeni, e verranno quindi impiegati tutti gli eventuali elementi figlio per marcare le singole parti di testo e tutti gli attributi necessari per codificarne le caratteristiche.

2. Pubblicazione delle risorse digitali

La seconda fase è incentrata sulla pubblicazione online delle risorse digitali create durante la prima fase, e quindi sulla creazione del portale web che ospiterà il centro di documentazione digitale e sull'implementazione di funzionalità pensate per la ricerca. Le API IIIF vengono elaborate da uno specifico *client* per generare un'interfaccia web in cui visualizzare e navigare le scansioni (IIIF Consortium, s.d., "Viewing objects"). È stato definito un *template* Jinja2 in cui viene inizializzata un'istanza di Mirador 3 — *viewer* IIIF che utilizza la nota libreria React — inserendo un riferimento allo URL in cui è archiviato il codice JavaScript del client. Nel *template* viene chiamata la specifica funzione `Mirador.viewer()` passando come parametro un oggetto JavaScript le cui proprietà servono a configurare il *viewer*; qui si inseriscono variabili che saranno estratte dinamicamente dalla richiesta HTTP. Infatti, nell'applicazione Flask si mappa uno URL con i parametri di percorso e identificatore a una funzione per il *renderig* del *template* Jinja2; questi parametri vengono estratti per costruire dinamicamente lo URL di un *Presentation* API, il *template* viene renderizzato passando lo URL come variabile e il *viewer* viene visualizzato nel browser (Helm, 2024, "Embedding Mirador"). Alla stessa funzione è stato mappato anche uno URL simile al precedente ma con l'aggiunta di un frammento che indica l'identificatore di una singola *canvas*; esso permetterà di caricare il *viewer* su una specifica *canvas* grazie alla funzione `Mirador.actions.setCanvas()` (*Ivi*, "Embedding Mirador"). Per visualizzare le trascrizioni TEI in pagine web, è stato programmato un complesso *style sheet* XSL in cui si inserisce la dichiarazione XML e l'elemento radice marcato con il tag `<xsl:stylesheet>`. Si definiscono *template* per trasformare gli elementi XML dati in input con elementi `<xsl:template>`: un *template* generale seleziona l'elemento radice per produrre un documento XHTML; *template* specifici selezionano i singoli elementi, anche con istruzioni condizionali. Il contenuto testuale di elementi e attributi viene così estratto, elaborato e aggiunto all'output XHTML. I file vengono trasformati mappando uno URL con i parametri di percorso e identificatore nell'applicazione Flask a una specifica funzione che analizza e trasforma in oggetti XML i file XML e lo *style sheet* XSL. Dopo di che, l'oggetto dello *style sheet* XSL viene convertito in un oggetto "trasformatore XSLT" e applicato all'oggetto del documento XML (Behnel, 2024, "XPath and XSLT with lxml"). Il documento trasformato in XHTML viene così inviato all'utente e la trascrizione visualizzata nel browser. Per permettere la navigazione del centro di documentazione digitale è stata creata una *homepage* per il portale da cui selezionare i documenti, disponibile online al seguente link: https://centrodocumentazione.fondazionebassetti.org/. Essa si basa su un *template* Jinja2 che

permette la generazione dinamica di schede descrittive dei documenti analizzando i file TEI ed estraendo da essi informazioni come titolo, autore, data, lingua e numero di pagine, oltre ai dati necessari per costruire gli URL per visualizzare *viewer* e trascrizioni. Nel momento di scrittura di questo *paper*, la *homepage* presenta quattro raggruppamenti tematici predefiniti per l'accesso ai documenti digitalizzati: pensiero europeista, Comune di Milano, Commissione Trilaterale e IPALMO. Successivamente verranno aggiunte delle funzionalità di ricerca avanzata per permettere all'utente di cercare documenti in base a titolo, autore, *range* di data di pubblicazione o parole chiave.
Il progetto prevede di implementare nelle regole di trasformazione dello *style sheet* XSL una marcatura semantica avanzata per il portale, utilizzando quando possibile gli elementi HTML che hanno uno specifico valore semantico, come `<article>`, `<aside>` e `<time>` (MDN Team, s.d.-a, "Semantics"); a questo si aggiunge la codifica di metadati strutturati tramite set di *microdata*, facendo principalmente riferimento al vocabolario di Schema.org, descrivendo entità e documenti con etichette come `Person`, `Organization` e `CreativeWork` (MDN Team, s.d.-b, "Microdata"). Ciò sarà utile per migliorare l'accessibilità del portale e permetterne l'utilizzo con strumenti assistivi, altro importante obiettivo del progetto per il quale verranno implementati anche gli attributi WAI-ARIA, come cui `role` (MDN Team, s.d.-c, "WAI-ARIA basics"). Su questi principi si è basata anche la programmazione dello *style sheet* CSS che definisce l'aspetto grafico del portale: i caratteri dei documenti sono stati uniformati con un font ad alta leggibilità di giusta grandezza, sono stati scelti colori che garantiscono un buon contrasto e sono stati evidenziati link, pulsanti e altri elementi con cui l'utente può interagire (*Ivi*, "CSS and JavaScript accessibility best practices"). È inoltre stata ricostruita nel centro di documentazione l'identità grafica dell'istituzione che conserva il fondo archivistico, modificando anche i colori e i font dell'interfaccia di Mirador 3 configurando l'istanza del *client* e inserendo uno *style sheet* CSS interno nel *template* Jinja2. Il progetto fa poi ampio uso di JavaScript per rendere più interessante l'aspetto delle pagine web, aumentare la possibilità di interazione dell'utente e aggiungere funzionalità al portale. Alcuni elementi, come quelli marcati dai *tag* `<note>` e `<stamp>` che codificano annotazioni a mano e timbri, sono stati inseriti all'interno di componenti espandibili dall'utente. Sono state programmate delle funzioni per passare da uno all'altro dei due elementi figlio degli elementi `<choice>` dei file TEI; essi servono a raccogliere insieme due proposte di codifica per la stessa porzione di testo, come abbreviazioni e forme sciolte, forme erronee e correzione ed espressioni originali e normalizzate (TEI Consortium, 2025, "Elements Available in All TEI Documents"). In maniera simile, è possibile passare da una all'altra delle varie letture di certe porzioni di testo che differiscono in versioni dello stesso documento, marcate da diversi elementi `<rdg>` all'interno dello stesso `<app>` (*Ivi*, "Critical Apparatus"). Le pagine con le trascrizioni sono poi dotate di barre laterali espandibili, tra cui una che riporta alcune delle informazioni contenute nel TEI *header* dei file TEI. Una seconda barra laterale contiene delle schede informative su persone e organizzazione menzionate nei documenti generate dinamicamente scaricando i metadati da Wikidata. Nell'elemento che marca il nome di queste entità è stato inserito il codice identificativo della relativa raccolta di metadati di Wikidata tramite l'attributo `ref`. L'identificativo viene riportato nella trascrizione trasformata, dalla quale viene estratto da una funzione JavaScript quando la pagina viene caricata e utilizzato nell'applicazione Python per costruire lo URL di *Linked Data Interface*; esso permette di scaricare i dati in formato JSON (Wikidata:Data access, 2025, "Linked Data Interface (URI)"), che infine sono utilizzati nella funzione JavaScript per generare le schede. Una terza barra laterale contiene delle opzioni per modificare le impostazioni di lettura, tra cui passare alla modalità notte o schermo intero, e mostrare i caratteri originali dei documenti. Altre opzioni permettono, per esempio, di posizionare il testo aggiunto a margine o sopra la linea nella sua posizione originale e mostrare il testo cancellato in originale o le componenti tipografiche delle pagine. Sono state create anche delle barre orizzontali espandibili poste nella parte superiore e inferiore del *viewport* che riportano intestazioni e piè di pagina originali, codificati con l'elemento `<fw>` e utilizzando rispettivamente i valori `header` e `footer` per l'attributo `type` (TEI Consortium, 2025, "Representation of Primary Sources"). Si prevede anche di implementare delle funzionalità pensate principalmente per i ricercatori. La prima consiste in un sistema per generare automaticamente citazioni bibliografiche, mentre la seconda riguarda l'annotazione di porzioni testuali in un'ottica di *social scholarship*.

3. Applicazione di sistemi AI

La terza e ultima fase si focalizza sull'implementazione di sistemi di intelligenza artificiale al centro di documentazione, puntando principalmente a utilizzare *large language model* (LLM) *open-source* in locale per ragioni economiche e di sicurezza dei dati. L'approccio principale prevede l'utilizzo delle API di piattaforme come Hugging Face e Ollama, che permettono di scaricare gli LLM sulla macchina e utilizzarne

le funzionalità all'interno del codice (Hugging Face Team, s.d., "Transformers"; Chiang & Morgan, s.d., "Quickstart"); si valuterà anche l'implementazione di LLM in maniera più complessa tramite *framework* come LangChain, LlamaIndex o Haystack. Verranno sviluppate delle funzionalità interattive per le pagine delle trascrizioni per permettere agli utenti di tradurre o sintetizzare dinamicamente porzioni testuali. Si valuterà poi la programmazione di una interfaccia per il recupero delle informazioni dai file TEI tramite meccanismi di *retrieval augmented generation* (RAG), una tecnica che permette agli LLM di arricchire l'output generato con dati recuperati da specifici set di documenti (Nagoudi et al., 2024, p. 22). Sarà di massimo interesse garantire l'estrazione della ricchezza semantica della codifica nei processi di generazione. Infine, si studierà la fattibilità di utilizzare i file TEI per il *fine-tuning*, cioè una tecnica di *transfer learning* con la quale si riaddestrano modelli preesistenti con nuovi dati (Zhang et al., 2023, "Fine-Tuning"), per permettere a un LLM di riconoscere e descrivere le componenti testuali in file di testo semplice; questo verrà fatto anche con l'obiettivo di automatizzare il processo di codifica di nuovi documenti.

BIBLIOGRAFIA

Appleby, M., Crane, T., Sanderson, R., Stroop, J. & Warner, S. (2020a, 3 giugno). Image API 3.0. IIIF | International Image Interoperability Framework. Consultato il 23 marzo 2025, da https://iiif.io/api/image/3.0/.

Appleby, M., Crane, T., Sanderson, R., Stroop, J. & Warner, S. (2020b, 3 giugno). Presentation API 3.0. IIIF | International Image Interoperability Framework. Consultato il 23 marzo 2025, da https://iiif.io/api/presentation/3.0/.

Behnel, S. (2024, 10 agosto). lxml - XML and HTML with Python. lxml. Consultato il 23 marzo 2025, da https://lxml.de/.

Chiang, M. & Morgan, J. (s.d.). Documentation. GitHub. Consultato il 23 marzo 2025, da https://github.com/ollama/ollama/blob/main/docs/README.md.

Ciula, A., Eide, Ø., Marras, C. & Sahle, P. (2023). Modelling Between Digital and Humanities, Open Book Publishers, Cambridge, https://doi.org/10.11647/OBP.0369.

Helm, L. (Ed.). (2024, 22 maggio). Mirador 3 Documentation. GitHub. Consultato il 23 marzo 2025, da https://github.com/projectmirador/mirador/wiki.

Hugging Face Team (s.d.). Documentations. Hugging Face. Consultato il 23 marzo 2025, da https://huggingface.co/docs.

IIIF Consortium (s.d.). How It Works. IIIF | International Image Interoperability Framework. Consultato il 23 marzo 2025, da https://iiif.io/get-started/how-iiif-works/.

MDN Team (s.d..-a). MDN Web Docs Glossary: Definitions of Web-related terms. MDN Web Docs. Consultato il 23 marzo 2025, da https://developer.mozilla.org/en-US/docs/Glossary.

MDN Team (s.d.-b). Web technology for developers. MDN Web Docs. Consultato il 23 marzo 2025, da https://developer.mozilla.org/en-US/docs/Web.

MDN Team (s.d.-c). Learn web development. MDN Web Docs. Consultato il 23 marzo 2025, da https://developer.mozilla.org/en-US/docs/Learn_web_development.

Meschini, F. (2020). Oltre il libro. Forme di testualità e digital humanities. Editrice Bibliografica.

Nagoudi, E. B., Alcoba Inciarte, A. & Muhammad, A. (2024, 27 maggio). Improving Archives-Focused LLMs with Retrieval Augmented Generation. In Duranti, L. & Rogers, C. (Eds), Artificial Intelligence and Documentary Heritage. SCEaR Newsletter, Special issue 2024. 21–26.

Pallets (2024, 13 novembre). Flask Documentation (3.1.x). Flask. Consultato il 23 marzo 2025, da https://flask.palletsprojects.com/en/stable/.

Sanderson R., Ciccarese, P. & Young, B. (2017, 23 febbraio). Web Annotation Data Model. W3C. Consultato il 23 marzo 2025, da https://www.w3.org/TR/annotation-model/.

Sechi, L. (2010). Editoria Digitale, Apogeo.

TEI Consortium (2025, 24 gennaio). P5: Guidelines for Electronic Text Encoding and Interchange. Text Encoding Initiative. Consultato il 23 marzo 2025, da https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html.

Wikidata:Data access. (2025, 4 gennaio). In Wikidata. Consultato il 23 marzo 2025, da https://www.wikidata.org/wiki/Wikidata:Data_access.

Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. (2023). Dive into Deep Learning. Cambridge University Press. Consultato il 23 marzo 2025, da https://d2l.ai/.