UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola
di Ingegneria

Corso di Laurea Magistrale
in
Ingegneria Informatica

Costruzione di un dataset di documenti scientifici per l'addestramento di una GNN

Candidato:
Alessandro Longo

Relatore:
Prof. Simone Marinai

Anno Accademico 2021-2022

"You can have data without information, but you cannot have information without data."
Daniel Keys Moran

Ringraziamenti

Ringrazio il professor Simone Marinai per l'aiuto e la guida datami nella stesura del lavoro di tesi, nonché alla pazienza e il tempo dedicatemi.
Un particolare ringraziamento va alla mia famiglia, a Lorena, a mia nonna Salvina e a tutti coloro che mi hanno supportato e sopportato nel mio percorso universitario.

Indice

Elenco delle figure

Elenco delle tabelle

Nomenclatura

Acronimi

GNN	Graph Neural Network
DLA	Document Layout Analysis
LCS	Longest Common Subsequence

Introduzione

Lo scopo di questa tesi è la presentazione di un algoritmo il cui scopo è la generazione di un dataset per l'addestramento di una GNN. Il dataset generato contiene coppie composte ognuna da un file LaTeX (.tex) e dal file pdf (.pdf) generato a partire da esso: oltre a questo vengono generati dei dati per il matching tra i due file, che forniscono le informazioni per associare ogni elemento presenti nel file di ouput (pdf) alla corrispondente porzione del file di input (LaTeX).

In questo modo, ognuno di essi può essere trattato come un nodo della GNN e le relazioni tra essi, come per esempio l'appartenenza alla stessa pagina o allo stesso blocco di testo, possono essere rappresentate con degli archi.

Struttura della tesi

Capitolo 1

Nel primo capitolo verrà introdotto il problema del DLA e alcune delle soluzioni che sono state impiegate nel corso degli anni per risolverlo.

Capitolo 2

Il secondo capitolo presenta una descrizione ad alto livello dell'algoritmo proposto e del suo processo di stesura, nonché la discussione di alcune scelte implementative e di possibili alternative.

Capitolo 3

L'ultimo capitolo analizza i risultati ottenuti, assesta la qualità del matching nelle coppie prodotte e suggerisce degli spunti per possibili miglioramenti futuri.

Capitolo 1

Il problema del DLA

1.1 Spiegazione del problema

Il problema affrontato in questa tesi è l'annotazione dei documenti scientifici, in particolare dell'Analisi del Layout dei Documenti, o DLA, ovvero l'identificazione e classificazione delle parti di un documento, come testi, immagini, tabelle, e altri, per codificarne la struttura logica dei contenuti.

1.2 I problemi dei metodi attuali

1.2.1 Dipendenza dall'apprendimento supervisionato

Il primo problema riscontrato nella DLA è che le metodologie attualmente utilizzate si basano fortemente sull'apprendimento supervisionato, il che rende il processo di raccolta dei dati molto lungo e dispendioso.

1.2.2 Disponibilità limitata dei documenti

Inoltre, non tutti i tipi di documenti sono accessibili pubblicamente a causa di problemi di policy, quindi molti dataset sono principalmente composti da articoli scientifici: questo limita la varietà di dati utilizzati per l'addestramento di modelli che risolvono la DLA e, di conseguenza, ne riduce le performance con documenti di altra natura.

1.2.3 Assenza del file sorgente

Anche i documenti liberamente accessibili online non sempre rendono disponibile anche il codice sorgente, per cui ci si trova a dover scegliere addestrare un modello con pochi dati etichettati manualmente o con molti non etichettati: nel primo caso il dataset ottenuto è molto ridotto e relativamente lento da costruire; nel secondo, invece, va a scapito della robustezza e affidabilità delle predizioni.

1.3 Soluzioni impiegate

1.3.1 Utilizzo di dati sintetici

Una soluzione per ovviare a questi problemi consiste nella generazione di un dataset sintetico, ovvero utilizzare per l'addestramento un insieme di documenti generati anch'essi artificialmente, e quindi già etichettati. Il vantaggio di questo metodo risiede nella quantità di dati disponibili, che può essere determinata a priori, e l'impiego di poche risorse; lo svantaggio, d'altra parte, è la variabilità ridotta e le basse performance con dati reali.

Capitolo 2

Il codice e l'algoritmo di matching

2.1 Linguaggio e risorse utilizzate

L'algoritmo che verrà ora presentato è stato scritto in Python 3.10 con l'aiuto di alcune librerie e moduli esterni.

2.2 Classi

Il codice presenta quattro classi distinte:

- La classe WebScraper, per l'ottenimento dei file dal database online Arxiv.
- La classe latexData, per il parsing e l'analisi del file LaTeX.
- La classe PDFData, per il parsing e l'analisi del file pdf.
- La classe MatchingTool, che contiene il vero e proprio algoritmo di matching.

2.2.1 WebScraper

La classe *WebScraper*, tramite la libreria *request*, accede all'archivio online di documenti scientifici Arxiv e tenta di effettuare il download delle coppie LaTeX-pdf per ogni articolo il cui codice identificativo rientra in un intervallo predeterminato dall'utente. Poiché il file sorgente LaTeX non è disponibile per tutti gli articoli nel database, vengono mantenute solo le coppie che presentano entrambi i file. Per questi articoli, la cartella compressa contenente i file sorgente viene estratta, il file sorgente principale viene individuato e, insieme al pdf, sono rinominati e organizzati in cartelle in modo da facilitare l'indicizzazione e il reperimento successivi.

2.2.2 LatexData

La classe *LatexData* si occupa di processare il file sorgente e di prepararne il contenuto per il matching. I passi includono:

- Sostituzione dei comandi "include" e "input" con il contenuto dei file chiamati

- Parsing e preprocessamento del documento
- Creazione di un albero gerarchico del contenuto
- Enumerazione e delle foglie di tale albero, corrispondenti ai singoli blocchi di contenuto (paragrafi, formule, tabelle, ecc.)

2.2.3 PDFData

La classe *PDFData* si occupa di processare, tramite l'utilizzo della libreria pdfminer, il file di output. Dopo aver estratto tutti gli elementi (linee di testo, immagini, ecc.), questi vengono salvati in una struttura dati che contiene, tra gli altri, il contenuto testuale, se sono di tipo *pdfminer.LTTextBox*, le coordinate della bounding box e la pagina di appartenenza.

2.2.4 MatchingTool

La classe *MatchingTool* è responsabile dell'accoppiamento tra i due file e contiene l'algoritmo vero e proprio. Questo si basa principalmente sull'algoritmo LCS, per la comparazione delle caselle di testo del pdf e le porzioni del file LaTeX.

2.3 L'algoritmo di matching

2.3.1

Capitolo 3

Presentazione e analisi dei risultati

3.1 Introduzione

Conclusioni

Bibliografia