

Progetto Finale Data and Document Mining: Algoritmo OCR per la classificazione di documenti

Bernardo Tiezzi

E-mail address

`bernardo.tiezzi@stud.unifi.it`

Abstract

La relazione seguente analizza le fasi realizzative del progetto d'esame di Data and Document Mining per il corso di Laurea Magistrale in Ingegneria Informatica dell'Università degli Studi di Firenze. L'idea alla base dello sviluppo è implementare un programma che emuli l'algoritmo OCR Text Recognition, al fine di ottenere una classificazione adeguata di varie pubblicazioni.

Future Distribution Permission

The author(s) of this report give permission for this document to be distributed to Unifi-affiliated students taking future courses.

1. Introduzione

I sistemi di riconoscimento ottico dei caratteri, detti anche **OCR**[1] (Optical Character Recognition), sono programmi dedicati al rilevamento dei caratteri contenuti in un documento e al loro trasferimento in testo digitale leggibile da una macchina. Di solito tale conversione viene effettuata da uno scanner. Il testo può essere convertito in formato **ASCII** semplice, **UNICODE** o, usufruendo di programmi più avanzati, in formati che tengono di conto l'impaginazione del testo stesso. L'OCR è dunque un campo di ricerca dell'AI legato al riconoscimento delle immagini.

Ad oggi esistono numerosi siti che raccolgono varie tipologie di pubblicazioni testuali digitalizzate: per ogni testo il sito consente all'utente di ottenerne un formato *PDF* oltre a fornirne una propria versione web. La maggioranza dei documenti non presenta tuttavia

una classificazione testuale adeguata, che ne semplifichi la comprensione.

L'idea alla base del progetto è dunque ovviare al disagio fornendo un algoritmo di tipo OCR Text Recognition, che possa essere utilizzato su tipologie differenti di documenti. Dato in input un testo digitale, il programma fornirà in output la classificazione concettuale del documento stesso, in formato PDF.

2. Obiettivi di progettazione

Per ottenere un programma in grado di eseguire correttamente la classificazione, sono stati ottenuti a-priori documenti in formato pdf, forniti dal sito [SpringerLink](#), assieme ai relativi file *HTML*, forniti dalla piattaforma. L'obiettivo è utilizzare i tag html della pagina web, che riportano la struttura del documento, in modo da individuare una corrispondenza testuale con il documento PDF ed ottenere la corretta classificazione.

3. Strumenti di Progettazione

Il progetto è stato implementato utilizzando il linguaggio di programmazione Python e l'IDE *Pycharm*. Le librerie principali, impiegate durante lo sviluppo del codice, sono:

- **BeautifulSoup**[2]: libreria utilizzata per leggere i file HTML ed ottenere i blocchi di testo, in base alla tipologia di tag HTML e alla relativa classe di appartenenza.
- **PyMuPDF**[3]: libreria impiegata per eseguire una scansione sui documenti PDF, for-

nendo in output un nuovo file che riporti i risultati ottenuti dal programma.

- **Scikit-Image**[4]: libreria necessaria per individuare la presenza di immagini all'interno del documento, non rilevabili attraverso l'utilizzo di PyMuPDF.

4. Descrizione dell'elaborato

Il programma è stato implementato come *python project*, suddiviso su più *file.py*. Di seguito sono descritti i file principali, andando ad analizzare le funzionalità di ognuno, necessarie all'esecuzione del software.

4.1. main.py

Il file **main.py** è il file principale del programma. Le librerie **BeautifulSoup** e **PyMuPDF** vengono incluse all'interno del sorgente. Inizialmente viene eseguito il download per il file HTML e PDF passando in input al programma l'indirizzo della pagina web (viene eseguito un controllo ogni volta per evitare di ottenere duplicati dei file). La piattaforma **SpringerLink** fornisce una banca dati contenente milioni di documenti, accessibili tramite account Springer o account istituzionale. Per ottenere i file viene utilizzato il server proxy, fornito dall'Università degli Studi di Firenze, assieme alle credenziali previste dalla modalità di autenticazione servizi SIAF.

```
with requests.Session() as s:
    s.proxies = {'https':
        'http://[Username]:[Password]
        @proxyunifi.unifi.it:8888'}

    response_link = s.post(link)
    r_link = s.get(link)
```

Il file HTML relativo al documento viene letto utilizzando la libreria *BeautifulSoup*, che consente di eseguire una ricerca all'interno del documento, filtrando i tag presenti in base alla tipologia e alla classe di appartenenza.

```
soup = BeautifulSoup(f.read(),
    'html.parser')
soup.find_all('type', class_='class')
```

Ogni blocco di testo selezionato consente di definire la struttura del documento. La scelta dei tag è ricaduta sulle seguenti tipologie:

- i *tag heading* per riconoscere l'intestazione e l'organizzazione del documento ipertestuale. La scelta si limita alle prime 4 tipologie più frequenti: da *h1* ad *h4*.
- le *equazioni* e formule matematiche presenti all'interno del file. I documenti analizzati sono per la maggior parte articoli scientifici
- i blocchi contenenti le descrizioni di *immagini* e *tabelle*
- i *referimenti bibliografici*.

Sfruttando la libreria **PyMuPDF** si effettuano modifiche al documento PDF.

```
import fitz
doc = fitz.open(f"{path_to_pdf_file}")
for page in doc:
    texts = page.get_text("json")
    texts = json.loads(texts)
    ...
```

Ogni pagina viene suddivisa in blocchi di testo, andando ad eseguire un riscontro con il contenuto di ogni tag selezionato. La classificazione viene eseguita disegnando un riquadro attorno ad ogni blocco di testo; se al termine dei controlli non si evidenzia nessuna appartenenza ad una classe semantica, verrà assegnata un'etichetta generica.

```
for block in texts['blocks']:
    ...
    page.draw_rect
    (Rect(block['bbox']),
    color=color_block, width=1.5)
```

Per individuare i titoli di paragrafi e sottoparagrafi è necessario controllare sia il font-weight del testo (Bold o Italics), sia la somiglianza tra la stringa selezionata nella pagina ed ogni stringa appartenente alla lista delle intestazioni. Se la percentuale di somiglianza risulta superiore ad un valore di soglia (90%), viene definita l'etichetta del blocco e si prosegue con l'iterazione successiva. Per individuare le equazioni il procedimento è analogo alla ricerca degli headers di testo; tuttavia la libreria **PyMuPDF** potrebbe non essere

in grado di riconoscere alcuni simboli matematici (\sum , \int , \pm etc.), oltre alle parentesi di sistema, disposte su più colonne. Per ovviare al problema vengono eseguiti ulteriori controlli, dividendo in blocchi l'equazione e confrontando ogni sequenza generata con il blocco di testo selezionato.

```
k = len(string.replace(" ", ""))
it_eq = 0
if k < len(eq.replace(" ", "")):
    while it_eq + k <= len(eq):
        if block['type'] == 0 and
           SequenceMatcher(None,
                             eq[it_eq:it_eq + k], s.replace(" ", "")).ratio() >= 0.6:
            ...
```

Una volta conclusi i restanti controlli per le classi scelte (descrizione immagini, tabelle e bibliografia), si effettua una verifica su casi particolari:

- il blocco di testo, pur non riconosciuto, è posto sulla stessa linea di un'equazione, rilevata in precedenza. Nel caso, essendo le formule matematiche disgiunte all'interno di un paragrafo, il blocco sarà parte dell'equazione stessa.
- se un'equazione non viene letta separatamente dalla libreria, è necessario ciclare sull'intera stringa, in modo da isolare le formule matematiche.

Nel caso in cui la pagina contenga un'immagine possono verificarsi due scenari differenti:

1. l'immagine viene rilevata da PyMuPDF
2. la libreria non riconosce l'immagine.

Nel primo caso viene regolarmente assegnata l'etichetta alla figura. Nel secondo caso viene utilizzato un approccio alternativo: il numero di immagini e descrizioni all'interno di una pagina del documento deve risultare lo stesso. In caso contrario si esegue un algoritmo RLSA, implementato all'interno del file *image.py* [15].

4.2. image.py

All'interno del file viene sviluppato un algoritmo alternativo per il rilevamento di figure all'interno di un documento.

```
find_image(BASE_PATH, DOC_NAME, idx,
           color_white, color_images):
```

La fase iniziale prevede di rimuovere dalla pagina PDF tutte le sequenze testuali e le figure individuate dalla libreria PyMuPDF. La rimozione avviene sovrascrivendo con un riquadro bianco varie sezioni del PDF.

```
for page in doc:
    ...
    for block in texts['blocks']:
        ...
        page2.draw_rect
        (Rect(block['bbox']),
         color=color_white, width=1.5,
         fill=color_white)
```

In questo modo, si ottiene la pagina del documento desiderata, contenente solo immagini. La pagina PDF viene trasformata in formato JPEG. Per poter lavorare sull'immagine viene eseguita una binarizzazione dei colori, utilizzando un valore di soglia arbitrario.

```
gray_painting = rgb2gray(painting)
binarized = gray_painting < 0.60
```

La presenza di alcuni "buchi" all'interno dell'immagine può tuttavia complicare la procedura di riconoscimento. Le operazioni morfologiche di *dilation* ed *erosion* consentono rispettivamente di coprire i buchi interni alla figura, mantenendo la sua forma originale.

```
square = np.array([
    [1, 1, 1],
    [1, 1, 1],
    [1, 1, 1]])

def multi_dil(im, num, element=square):
    ...
def multi_ero(im, num, element=square):
    ...
multi_dilated = multi_dil(binazed, 7)
area_closed = area_closing
(multi_dilated, 50000)
multi_eroded = multi_ero(area_closed, 7)
```

La fase di classificazione prevede di assegnare la stessa etichetta a gruppi di pixels adiacenti, considerati appartenenti alla stessa regione. Successivamente è possibile misurare le caratteristiche di ogni regione sfruttando un'apposita funzione; alcuni esempi sono l'area, il perimetro e la lunghezza degli assi. Queste caratteristiche risultano utili per eseguire un filtraggio sulle regioni: quest'ultime infatti potrebbero includere del rumore.



```
label_im = label(opened)
regions = regionprops(label_im)
for num, x in enumerate(regions):
    area = x.area
    if area > 50:
        ...
```






Le regioni che presentano un'area inferiore ad un minimo stabilito risultano inutili, così come quelle il cui rapporto tra le lunghezze degli assi è sbilanciato. È possibile evidenziare le regioni di interesse generando una maschera che nasconda quelle non necessarie dell'immagine. Utilizzando i *bounding_box* di ciascuna regione evidenziata, si individua il perimetro dell'intera immagine, che viene restituito dalla funzione.

```
minr, minc, maxr, maxc = props.bbox
...
box_image[0] = xmin
box_image[1] = ymin
box_image[2] = xmax
box_image[3] = ymax
return box_image
```

5. Testing

La fase di testing è stata eseguita su un dataset di 10 documenti, scaricati dalla piattaforma **SpringerLink**. Per ogni documento si è effettuato una ricerca di varie tipologie di testo all'interno del file HTML, fornendo in output un file PDF, dove sono evidenziati, tramite riquadri di colore differente, sequenze testuali ed immagini con l'etichetta della classe di appartenenza. Di seguito vengono mostrate le classi selezionate per testare il programma.

- Intestazioni 
- Descrizione Immagini 

- Descrizione Tabelle 
- Immagini 
- Formule Matematiche 
- Riferimenti bibliografici 
- Testo generico 

Al fianco di ogni classe è posto il colore che la caratterizza. I controlli riguardo la correttezza dei test vengono effettuati visivamente, valutando di volta in volta la classificazione del documento. I risultati sono riportati in tabella 1. In tabella 2

Classi	Docs	Test/Class	% Correttezza
Intestazioni	10	195	99.48%
Descrizione Immagini	10	66	98.48%
Descrizione Tabelle	10	66	98.48%
Immagini	10	77	94.81%
Formule Matematiche	10	182	76.92%
Riferimenti Bibliografici	10	386	100%

Table 1: Lista delle classi selezionate, con dimensione del dataset di documenti utilizzato, numero di test per classe e percentuale di correttezza dell'algoritmo

sono riportati i risultati dei test condotti su ogni classe per ogni documento PDF analizzato. In tabella 3 sono elencati gli articoli utilizzati durante la fase di testing.

5.1. Analisi dei Risultati

Osservando la tabella 2 a pagina 8 si può notare come le percentuali di correttezza siano molto alte per alcune classi, per esempio i riferimenti bibliografici ottengono sempre un riscontro del 100%, e siano più basse per classi come le equazioni matematiche. Consideriamo la prima riga della tabella: la classificazione per le intestazioni presenti nell'articolo è corretta al 95%. Tale risultato è dovuto alla presenza di simboli matematici all'interno del titolo di un paragrafo. Poichè la libreria **PyMuPDF** non è in grado di riconoscere tali caratteri, la somiglianza tra il blocco di testo contenuto nel PDF e l'istanza corrispondente contenuta nel file HTML è inferiore al valore di soglia utilizzato

(0.9). Ragionamento analogo è valido per la classificazione delle *figure caption*: nell'articolo 6 non viene rilevata la presenza di una didascalia a causa di un indice di somiglianza inferiore alla soglia definita. L'articolo 7 non riconosce una *table caption*: all'interno dell'articolo una tabella viene riportata su più pagine del documento; nonostante la tabella presenti una didascalia su ogni pagina, nel file HTML questo non viene rilevato.

Immagini. La classificazione delle figure, come spiegato in fondo al paragrafo 4.1, può essere eseguita tramite due metodi differenti:

- suddivisione in blocchi tramite **PyMuPDF**
- recupero del *bounding box* tramite la funzione `find_image()`.

Per evitare di dover procedere all'analisi di ogni singola pagina del documento tramite la funzione `find_image`, si effettua una ricerca delle didascalie per le figure su ogni pagina del PDF. Nel caso sia individuata una *caption*, senza l'immagine corrispondente, si esegue l'algoritmo. Dall'analisi dei risultati ottenuti si può notare come, nonostante la classificazione delle immagini presenti un grado di accuratezza molto elevato (in media 94.48%), siano presenti alcuni errori di valutazione:

- la classificazione delle figure dipende dalla capacità di riconoscere la presenza di *caption* all'interno della pagina; in caso contrario la funzione `find_image` non sarà eseguita.
- Nel caso un'immagine presenti dei blocchi di testo (ad esempio l'immagine di una tabella), PyMuPDF potrebbe rilevare tali caratteri indipendentemente dalla figura. Rimuovendo il testo, l'algoritmo eseguirà l'analisi su un'immagine incompleta.
- La presenza di eventuali componenti del PDF non riconosciute e non appartenenti ad una figura specifica può generare rumore modificando le dimensioni del *bounding box*.

Di seguito vengono mostrati due esempi di classificazione per immagini.

22 K. Zhou et al.

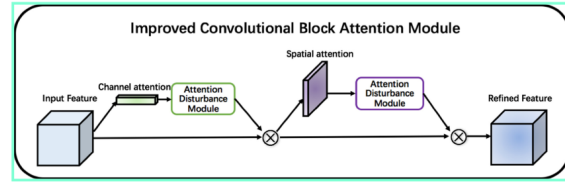


Fig. 4. The overview of ICBAM. We add attention disturbance module behind the channel and spatial attention in the training process to obtain more robust attention features at every convolutional block of deep networks. The structures of these two attention disturbance modules are identical.

Figure 1: Classificazione di immagini eseguita correttamente

1 C. Cheng et al.

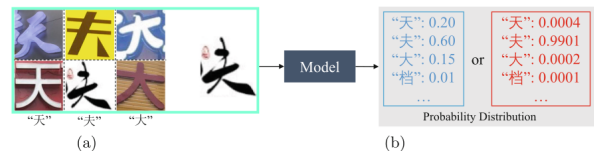


Fig. 1. Example of Chinese characters and 2 kinds of model prediction. (a) The 2 images in each column are a same character class and the 3 in each row are different. It shows the high inter-class similarity and the large intra-class variance which is a fine-grained attribute. (b) Although the 2 different probability distributions have a same prediction in training set, the left one holds a higher entropy that can describe the learned feature better. Obviously, “天” is far more similar to “夫” than “档”, so the confidences on them is supposed to have a large distinction.

Figure 2: Classificazione per immagini errata

In Fig.1 la classificazione dell'immagine è corretta; in Fig.2 la libreria PyMuPDF rileva la presenza della parte sinistra della figura ma non la parte destra. La funzione `find_image` non viene chiamata, poichè la presenza della didascalia sottostante porta il programma a considerare l'intera figura come classificata correttamente.

Equazioni. La classificazione delle equazioni avviene tramite un processo analogo a quella svolta per gli altri blocchi di testo. La differenza sta nella maggiore complessità del processo. I simboli matematici sono difficilmente letti correttamente dal programma che sostituisce il simbolo con un carattere UNICODE; questo produce ambiguità e non consente di classificare correttamente la formula matematica. Eseguire

test di somiglianza tra sottosezioni delle formule può migliorare la correttezza del programma. Le immagini seguenti mostrano i risultati ottenuti.

3.1 Review of Cross-Entropy Loss

We first review the common operation in a classification problem. Given an input sample x with label y , a classification model produces C scores $\{z_i\}_{i=1}^C$. Then we canonically get the output probability distribution \mathbf{p} by softmax function:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (1)$$

The derivative of softmax is:

$$\frac{\partial p_i}{\partial z_j} = \begin{cases} p_j(1 - p_j), & i = j \\ -p_i p_j, & i \neq j \end{cases} \quad (2)$$

The cross-entropy (CE) loss and its derivative are:

$$L_{CE} = -\log p_y \quad (3)$$

$$\frac{\partial L_{CE}}{\partial z_i} = \begin{cases} p_i - 1 < 0, & i = y \\ p_i > 0, & i \neq y \end{cases} \quad (4)$$

Figure 3: Classificazione di equazioni eseguita correttamente

Modeling Right-skewed.

$$h(t; \theta) = \frac{\left(\frac{\lambda}{2\mu t}\right)^{\frac{1}{\mu}} \exp\left\{-\frac{\lambda(t-\mu)^2}{2\mu^3 t}\right\}}{\Phi\left(-\sqrt{\frac{2}{t}}\left(\frac{t}{\mu}-1\right)\right) - \exp\left(\frac{2\lambda}{\mu}\right) \Phi\left(-\sqrt{\frac{2}{t}}\left(\frac{t}{\mu}+1\right)\right)}$$

where $\phi(t)$ is the probability density function of the normal distribution and $\Phi(t)$ is the normal cumulative distribution function.

Log-logistic Distribution

The log-logistic distribution is the probability distribution of a random variable whose logarithm has a logistic distribution. It is similar in shape to the log-normal distribution but has heavier tails. The log-logistic distribution has a flexible functional form, and it is one of the survival time models in which the hazard rate may be decreasing, increasing, or hump-shaped, meaning it initially increases and then decreases. For $\theta = (\alpha, \beta)' \in \mathbb{R}^+ \times \mathbb{R}^+$, the survival and hazard functions of log-logistic distribution are expressed as

Figure 4: Classificazione per equazioni errata

In Fig.3 le 4 equazioni sono classificate correttamente mentre in Fig. 4, la presenza di simboli sconosciuti, porta il programma a classificare contemporaneamente il blocco di testo come **equazione** e **testo generico**.

Le **references**, per qualunque documento, vengono catalogate con un'accuratezza del 100%; questo risultato è ottenuto grazie alla loro posizione all'interno del documento. I riferimenti bibliografici sono sempre posti al termine di un articolo, all'interno dello stesso paragrafo; questo aspetto semplifica il riconoscimento dei blocchi di testo.

6. Conclusioni Finali

L'analisi condotta ha consentito di analizzare accuratamente il funzionamento di un algoritmo di tipo OCR per il riconoscimento dei caratteri e la classificazione di documenti. Il programma descritto consente dunque, partendo da una collezione di testi in formato digitale, di ricavare informazioni riguardo la struttura interna dei vari articoli e la presenza o meno di alcune tipologie di testo. Individuare le intestazioni può facilitare la lettura e lo scorrimento dei file, andando a sviluppare un software che fornisca indici interattivi, in base alla suddivisione in blocchi. La presenza di formule matematiche consente di filtrare, all'interno di una banca dati digitale, gli articoli di natura scientifica. Ulteriori progetti di sviluppo dovrebbero concentrarsi sul migliorare ulteriormente l'accuratezza del programma, scegliendo un valore di soglia ottimale per l'analisi della somiglianza tra i blocchi di testo, in modo da bilanciare eventuali falsi positivi e negativi. Inoltre è possibile individuare un numero superiore di classi, in modo da ricavare maggiori informazioni e catalogare più tipologie di documenti (legali, economico-finanziari o di altra natura).

References

- [1] Riconoscimento ottico dei caratteri. https://it.wikipedia.org/wiki/Riconoscimento_ottico_dei_caratteri. 1
- [2] BeautifulSoup. <https://www.crummy.com/software/BeautifulSoup/>. 1
- [3] Pymupdf. <https://github.com/pymupdf/PyMuPDF>. 1
- [4] Image processing in python. <https://scikit-image.org/>. 2
- [5] Maximum entropy regularization and chinese text recognition. *Document Analysis Systems*, 1:3–17, 2020. 8
- [6] An improved convolutional block attention module for chinese character recognition. *Document Analysis Systems*, 2:18–29, 2020. 8
- [7] Classification of phonetic characters by space-filling curves. *Document Analysis Systems*, 7:89–100, 2020. 8
- [8] High performance offline handwritten chinese text recognition with a new data preprocessing and augmentation pipeline. *Document Analysis Systems*, 4:45–59, 2020. 8
- [9] Von neumann stability analysis of dg-like and pnpn-like schemes for pdes with globally curl-preserving evolution of

- vector fields. *Communications on Applied Mathematics and Computation*, 4:945–985, 2022. 8
- [10] Adapting ocr with limited supervision. *Document Analysis Systems*, 3:30–44, 2020. 8
 - [11] Alec: An accurate, light and efficient network for captcha recognition. *Document Analysis Systems*, 5:60–73, 2020. 8
 - [12] p-multilevel preconditioners for hho discretizations of the stokes equations with static condensation. *Communications on Applied Mathematics and Computation*, 4:783–822, 2022. 8
 - [13] Modeling right-skewed heavy-tail right-censored survival data with application to hiv viral load. *Bulletin of the Malaysian Mathematical Sciences Society*, 2022. 8
 - [14] The benefits of close-domain fine-tuning for table detection in document images. *Document Analysis Systems*, 15:199–215, 2020. 8
 - [15] J. Manansala. Image processing with python: Connected components and region labeling. <https://medium.com/swlh/image-processing-with-python-connected-components-and-region-labeling-3eef1864b951>, 2021. 3

Articles	Heading		Fig Caption		Table Caption		Image		Equation		References	
	Tot	Correct	Tot	Correct	Tot	Correct	Tot	Correct	Tot	Correct	Tot	Correct
Article 1	20	95%	4	100%	5	100%	5	80%	16	93.75%	40	100%
Article 2	13	100%	7	100%	4	100%	8	100%	2	100%	23	100%
Article 3	13	100%	8	100%	4	100%	9	100%	1	100%	15	100%
Article 4	21	100%	3	100%	4	100%	4	100%	5	80%	25	100%
Article 5	20	100%	14	100%	12	100%	15	93.34%	50	78%	48	100%
Article 6	16	100%	5	80%	8	100%	6	100%	5	100%	32	100%
Article 7	14	100%	8	100%	2	100%	9	88.89%	5	80%	31	100%
Article 8	47	100%	6	100%	18	100%	6	100%	68	63.24%	55	100%
Article 9	17	100%	9	100%	4	75%	12	91.67%	24	91.67%	79	100%
Article 10	14	100%	2	100%	5	100%	3	100%	6	83.34%	38	100%

Table 2: Tabella percentuali per documento

Articles	Documents	
Article 1	Maximum Entropy Regularization and Chinese Text Recognition [5]	
Article 2	An Improved Convolutional Block Attention Module for Chinese Character Recognition [6]	
Article 3	Classification of Phonetic Characters by Space-Filling Curves [7]	
Article 4	High Performance Offline Handwritten Chinese Text Recognition with a New Data Preprocessing [8]	
Article 5	Von Neumann Stability Analysis of DG-Like and PNPm-Like Schemes for PDEs with Globally Curl-Preserving Evolution of Vector Fields [9]	
Article 6	Adapting OCR with Limited Supervision [10]	
Article 7	ALEC: An Accurate, Light and Efficient Network for CAPTCHA Recognition [11]	
Article 8	p-Multilevel Preconditioners for HHO Discretizations of the Stokes Equations with Static Condensation [12]	
Article 9	Modeling Right-skewed Heavy-tail Right-censored Survival Data with Application to HIV Viral Load [13]	
Article 10	The Benefits of Close-Domain Fine-Tuning for Table Detection in Document Images [14]	

Table 3: Articoli utilizzati durante la fase di testing