Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Triennale in Ingegneria Informatica

Ricerca di grafemi in manoscritti

Candidato
Marco Benelli

Relatore
Prof. Simone Marinai

Anno Accademico 2019/2020

Indice

Introduzione

Negli ultimi anni siamo andati verso l'automatizzazione di sempre più compiti, dai più semplici ai più complessi e con vari gradi di successo. Alcuni di questi compiti vengono lasciati ai computer perché troppo complessi per gli umani; altri invece risultano elementari alle persone, ma si incontrano difficoltà nell'implementarli in maniera automatica. Si potrebbe quindi pensare che sia illogico cercare di automatizzare qualcosa che le persone trovano naturale, ma si sbaglierebbe. Infatti i computer hanno numerosi vantaggi rispetto alle persone: sono più veloci per le operazioni basilari e il loro tempo ha meno valore.

Una delle operazioni che alle persone riesce in maniera naturale è la lettura di documenti a partire da immagini degli stessi. Per un programma questo è un compito molto difficile, basti pensare ai CAPTCHA che si trovavano maggiormente fino a pochi anni fa.

Per questo progetto lo scopo non è la lettura, ma solamente il riconoscimento di alcuni grafemi in immagini di manoscritti medievali. Un grafema è un segno grafico a cui è associata una lettera; questa associazione non è biunivoca, infatti più grafemi possono corrispondere alla stessa lettera se quella lettera si può scrivere in più modi.

Questo può risultare utile perché, conoscendo l'evoluzione della calligrafia nel corso dei secoli, il diverso stile di scrittura di una lettera ci può aiutare

nella datazione dei manoscritti.

Il programma è scritto in *Python* e le librerie che sono state usate sono: *OpenCV* [1] per le operazioni sulle immagini; *numpy* [2, 3] per le operazioni sulle array; *matplotlib* [4] per visualizzare alcuni grafici; *scipy* [5] per alcune operazioni particolari.

Ciò che faremo per raggiungere il nostro obiettivo è, a partire da un grafema di riferimento, estrarre dal testo quelli simili. Per l'estrazione dei caratteri si useranno proiezioni ortogonali e componenti connesse, e per valutare la somiglianza, invece, sarà impiegato l'indice di Jaccard. Vedremo poi come poter introdurre dell'interattività nel programma facendo scegliere all'utente se una determinata immagine rappresenta il grafema desiderato. Per tutte queste modalità saranno fatti degli esperimenti per valutarne l'efficienza.

Capitolo 1

Approccio proposto

L'approccio che è stato utilizzato si articola in più fasi che andremo a vedere nel corso di questo capitolo.

1.1 Preprocessing

Le immagini di partenza sono lette e convertite in immagini in scala di grigi usando *OpenCV* (gli scan di partenza sono a colori).

Successivamente vengono binarizzate, ovvero passano da essere in scala di grigi ad essere in bianco e nero. Per la binarizzazione viene usata la soglia di 128, quindi tutti i pixel che hanno un valore < 128 diventano neri, mentre quelli con un valore ≥ 128 diventano bianchi.

Il valore di soglia (128), può chiaramente essere cambiato. È stato scelto il valore di 128 poiché a metà fra 0 e 255, che è il massimo valore che può avere un pixel in scala di grigi nella risoluzione a 8 bit per ogni pixel. Inoltre il valore di 128 sembra funzionare bene per le immagini di nostro interesse.

Se si dovessero analizzare altri scan, si può aggiustare il valore di soglia o addirittura cambiare il metodo di binarizzazione usandone uno più raffinato.

1.2 Segmentazione

Per segmentazione si intende la separazione del documento nei vari caratteri che lo compongono. Questo processo non può essere perfetto se non si conosce la lingua che si vuole leggere, poiché in certi casi nella scrittura a mano possono esserci dei tratti ambigui che si riescono a capire solamente guardando il contesto della parola o addirittura della frase.

1.2.1 Separazione delle righe

Prima di separare i caratteri di una parola dobbiamo separare il documento nelle righe di testo che lo compongono. Per fare ciò, possiamo contare, in ciascuna riga di pixel, quanti sono neri e prendere i minimi relativi come punti di separazione. Nella figura 1.1 è riportato il grafico del numero di pixel neri delle righe in una pagina.

Guardando attentamente la figura 1.1, si nota che i minimi relativi del grafico sono molti di più delle righe di testo e, in particolare, molti minimi vanno a cascare nel mezzo di una riga. Questo è un problema che possiamo risolvere con delle finestre scorrevoli.

Media mobile

Per calcolare la media mobile possiamo utilizzare una finestra scorrevole. Applicare una finestra scorrevole di grandezza n ad un array di numeri, significa sostituire ad ogni valore, la media degli n valori più vicini. Questo processo tende a far diminuire la differenza fra un elemento e il successivo e porta ad eliminare molti massimi e minimi relativi.

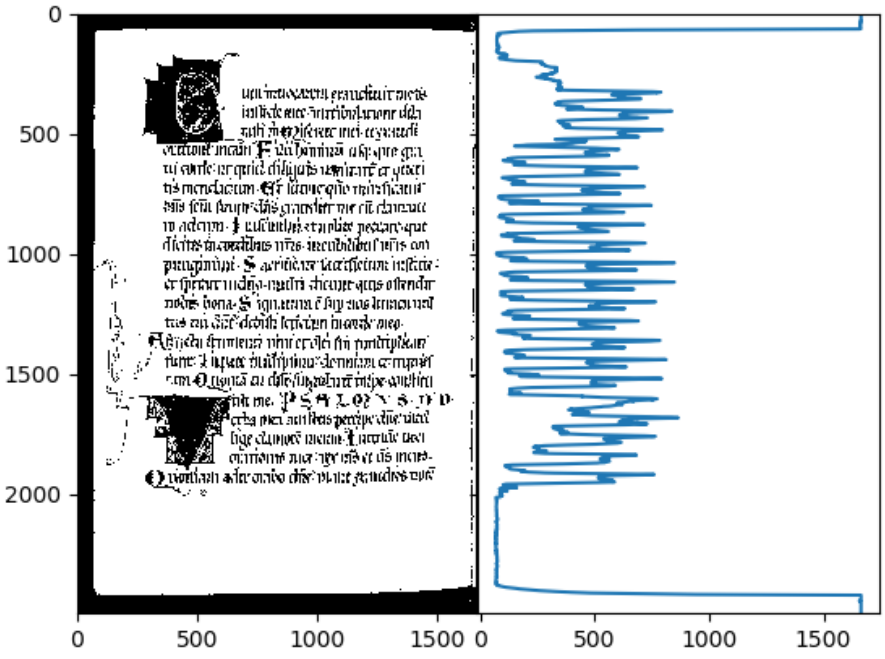

Figura 1.1: A sinistra l'immagine binarizzata, a destra il conto dei pixel neri in ciascuna riga dell'immagine (sulle ordinate il numero della riga e sulle ascisse il numero di pixel).

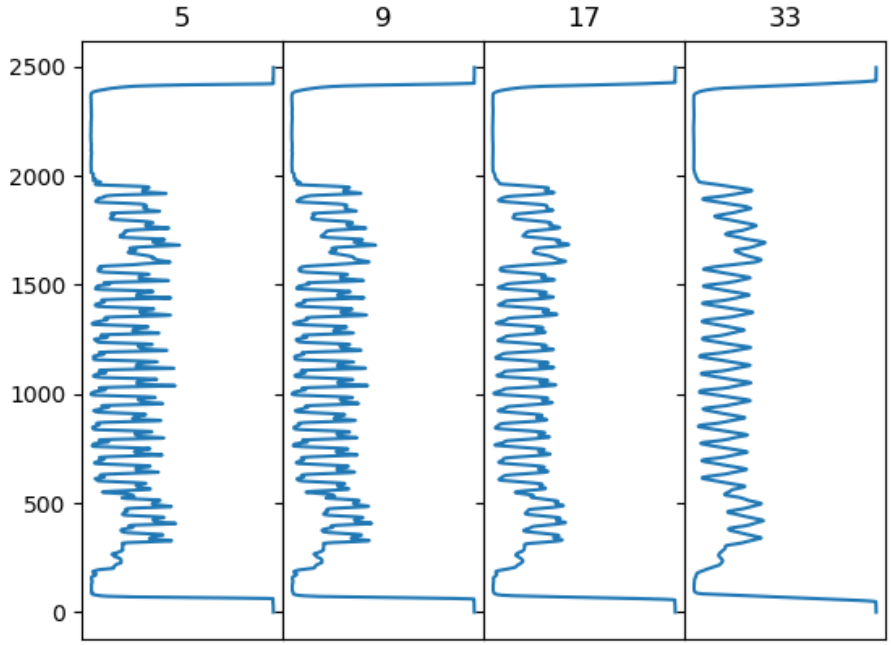

Figura 1.2: I grafici relativi a finestre scorrevoli di varia grandezza (sopra ciascun grafico è segnata la dimensione della finestra).

Nella figura 1.2 possiamo vedere come la finestra scorrevole agisce sull'array di partenza. In particolare in figura 1.3 possiamo vedere come i minimi relativi adesso si trovano sempre fra due righe di testo.

1.2.2 Separazione dei caratteri

Nonostante non si possa sperare di separare i caratteri con esattezza, possiamo comunque separare le varie righe e fare delle ipotesi sui punti in cui finisce un carattere e inizia il successivo usando lo stesso metodo visto per separare le righe.

In figura 1.4 si vede una riga di testo e la proiezione verticale corrispondente ad una finestra scorrevole di dimensione 9; mentre nella figura 1.5 è

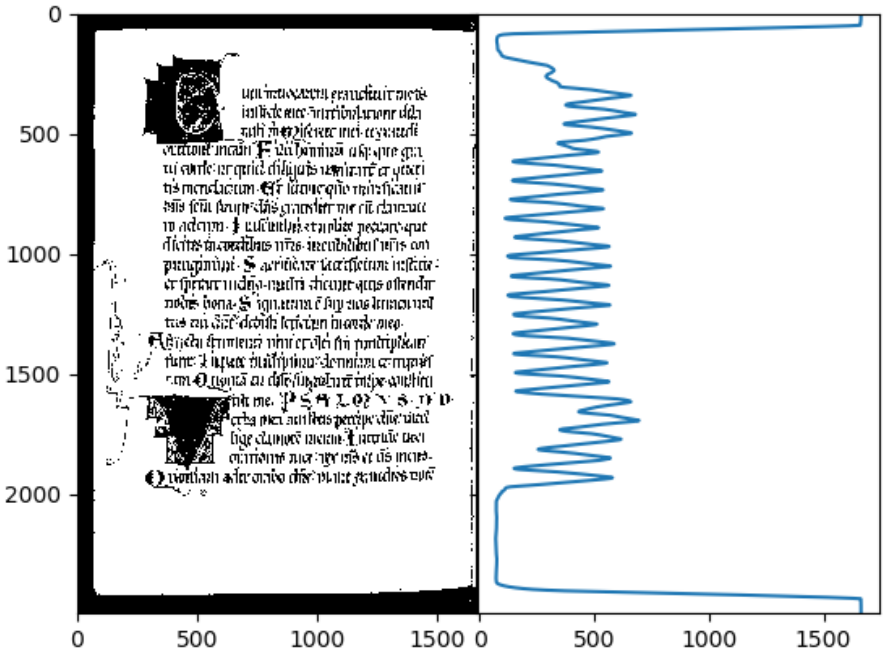

Figura 1.3: L'immagine di partenza affiancata alla proiezione a cui è stata applicata una finestra scorrevole di 33.

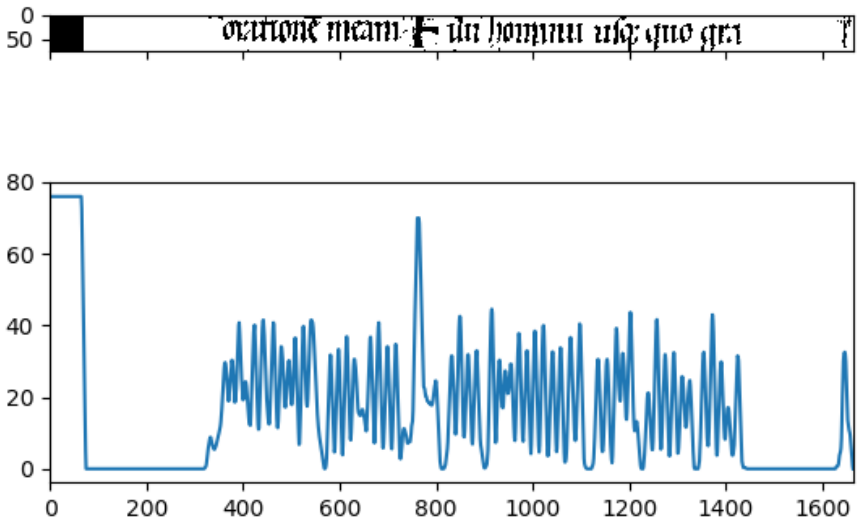

Figura 1.4: Sopra la riga di testo e sotto il grafico della proiezione verticale a cui è stata applicata una finestra scorrevole di dimensione 9.

Figura 1.5: Una riga di testo sezionata verticalmente.

presente la stessa riga con delle barre verticali in corrispondenza dei minimi relativi.

Come si vede dalla figura 1.5 e come avevamo predetto all'inizio del capitolo, la separazione dei caratteri non è stata perfetta. Nonostante ciò, i corretti punti di divisione sono tutti stati individuati, l'unico problema è che sono stati individuati anche punti di divisione incorretti. Questo non sarà comunque un problema come vedremo nel capitolo successivo.

Figura 1.6: Un rettangolo dell'immagine contenente la lettera *d*.

1.3 Estrazione delle lettere

Per estrarre le lettere da una riga di testo come quella in figura 1.5, dobbiamo guardare non solo i rettangoli di pixel che sono delimitati da due minimi nella proiezione verticale, ma dobbiamo anche guardare gruppi di 2 o 3 rettangoli contigui. Molte lettere, infatti, sono separate da 1 linea verticale e alcune, come la *m*, anche da 2.

Adesso che abbiamo una porzione di immagine rettangolare come quella in figura 1.6, dobbiamo estrarre la lettera vera e propria. Infatti, si possono vedere dei pezzi delle lettere confinanti che sono entrati nel rettangolo corrispondente alla nostra lettera.

1.3.1 Componenti connesse

Per prima cosa dobbiamo individuare le componenti connesse dell'immagine. Consideriamo un grafo non orientato che ha come vertici i pixel neri e come archi quelli andiamo adesso a definire. Possiamo dire che esiste un arco fra due vertici, se i pixel (poligoni) corrispondenti condividono un lato (connessione a 4) oppure se condividono un lato o un vertice (connessione a 8). La figura 1.7 mostra i due diversi tipi di connessione.

Nel nostro caso utilizzeremo la connessione a 4, ma dobbiamo tenere a mente che abbiamo sempre l'opzione della connessione a 8, che per altri scan

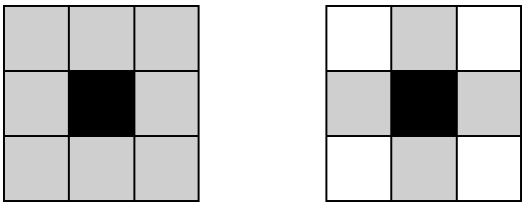

Figura 1.7: I due tipi di connessione. A destra la connessione a 4 e a sinistra la connessione a 8.

Figura 1.8: La lettera al termine dell'estrazione.

potrebbe essere più favorevole.

1.3.2 Procedura di estrazione

Una volta individuate le componenti connesse, la procedura di estrazione consiste nel prendere la componente connessa che conta il maggior numero di vertici (pixel neri) e rendere bianchi i pixel delle altre componenti. Infine, l'ultimo passaggio è ritagliare l'immagine orizzontalmente e verticalmente in modo da non avere righe o colonne che contengano solo pixel bianchi.

Applicando anche questi ultimi passaggi, il risultato è quello di figura 1.8.

1.4 Confronto fra lettere

Questa è la fase più critica dell'intero processo. Si tratta della fase di confronto fra due lettere per vedere se sono la stessa lettera. Visto che è una procedura complessa, deve essere suddivisa in procedure più piccole.

1.4.1 Ridimensionamento

Non è scontato (in generale è improbabile) che le due lettere che vogliamo confrontare siano delle stesse dimensioni, ma per poterle confrontare vorremmo che lo fossero.

OpenCV, così come molte altre librerie, ci offre una funzione per ridimensionare un'immagine. Quindi un'implementazione banale potrebbe essere quella di ridimensionare una delle immagini per renderla della stessa dimensione dell'altra. Questo, tuttavia comporterebbe una perdita delle proporzioni originali, e quindi non potrà andare bene visto che la forma di una lettera così come la intendiamo dipende strettamente dalle sue proporzioni.

Metodo centrale

Un altro metodo semplice, ma sicuramente più intelligente consiste nell'aggiungere colonne bianche all'immagine più stretta in quantità uguale a destra e a sinistra e, successivamente, aggiungere righe bianche a quella più bassa (scelta fra quella che prima era la più larga e quella ottenuta dall'aggiunta di colonne alla più stretta), ancora una volta in quantità uguale sopra e sotto. Chiaramente i passaggi possono essere scambiati, ovvero possiamo anche prima le righe e poi le colonne. Inutile dire che se le righe o le colonne da aggiungere sono in numero dispari, si dovrà dare una preferenza ad una delle due direzioni, ma se le immagini hanno una risoluzione abbastanza elevata, ci si può aspettare che questa scelta non porti a differenze sostanziali. Nel corso del testo ci si riferirà a questo metodo chiamandolo il metodo centrale o metodo statico, visto che le immagini vengono confrontate allineandone i centri senza spostarli.

Metodo proporzionale

Un altro metodo, che in particolari situazioni può risultare più efficace, potrebbe essere quello di ridimensionare proporzionalmente l'immagine più stretta e, in un secondo momento, aggiungere righe bianche a quella più bassa. Come prima, anche in questo caso possiamo invertire le dimensioni, ovvero ridimensionare la più bassa e poi aggiungere righe alla più stretta. Come si può intuire questo metodo è più efficace quando ci sono lettere di diversa dimensione, ma con lo stesso stile di scrittura, visto che si ha un ridimensionamento proporzionale di una delle lettere.

Metodo del baricentro

Si tratta di una variazione al metodo centrale. La differenza sostanziale è che con questo metodo, invece di allineare i centri, si allineano i baricentri. Questo comporta che le righe e le colonne vengano aggiunte in due passaggi: si aggiungono prima righe in alto all'immagine che sporge meno verso l'alto, e si aggiungono righe in basso all'immagine che sporge meno verso il basso (che non necessariamente sarà la stessa immagine di prima). Verrà fatto un procedimento analogo per le colonne, guardando indipendentemente a destra e a sinistra.

Ricordiamo come si calcolano le coordinate del baricentro (o del centro di massa). In ogni immagine associamo un peso pari a 1 ai pixel neri e un peso pari a 0 ai pixel bianchi. Se consideriamo un'immagine A con m righe e n colonne e indichiamo con A^i_j l'elemento di A alla riga i-esima e alla colonna j-esima, il baricentro si calcola nel modo seguente:

$$\left(\frac{\sum_{0 \le i < m} \sum_{0 \le j < n} i A_j^i}{\sum_{0 \le i < m} \sum_{0 \le j < n} A_j^i} \quad , \quad \frac{\sum_{0 \le i < m} \sum_{0 \le j < n} j A_j^i}{\sum_{0 \le i < m} \sum_{0 \le j < n} A_j^i} \right)$$

Dove la prima coordinata si riferisce alla riga e la seconda alla colonna.

Per semplificare la formula, da ora in avanti daremo per scontato che l'indice i va da 0 a m e l'indice j da 0 a n. Adoperando questo abuso di notazione e un'ulteriore semplificazione, possiamo indicare il baricentro come segue.

$$\left(\frac{\sum_i i \sum_j A_j^i}{\sum_{i,j} A_j^i} \quad , \quad \frac{\sum_j j \sum_i A_j^i}{\sum_{i,j} A_j^i} \right)$$

Portando fuori il denominatore otteniamo infine:

$$\frac{1}{\sum_{i,j} A_j^i} \left(\sum_i i \sum_j A_j^i \quad , \quad \sum_j j \sum_i A_j^i \right)$$

Metodo del momento di inerzia

Questo metodo è un'evoluzione del metodo del baricentro. Infatti si comporta in maniera simile a questo, ma aggiungendo un passaggio ulteriore. Una volta allineate le immagini con i corrispettivi baricentri, si scala l'immagine con il momento di inerzia minore, in modo che abbiano entrambe lo stesso momento di inerzia. A questo punto si procede come prima per rendere le immagini delle stesse dimensioni.

Prima di mostrare il calcolo del momento di inerzia, definiamo altre notazioni per semplificare le formule.

$$s(A) := \sum_{i,j} A^i_j$$

$$s^i(A) := \sum_{j} A^i_j$$

$$s_j(A) := \sum_{i} A^i_j$$

Ora che abbiamo queste tre definizioni possiamo semplificare ulteriormente la formula del baricentro che avevamo trovato in precedenza in questo modo:

$$\frac{1}{s(A)} \left(\sum_i i s^i(A) \quad , \quad \sum_j j s_j(A) \right)$$

Vediamo adesso come calcolare il momento di inerzia. La formula di partenza è questa:

$$\sum_{i,j} A^i_j \left((i - \tilde{\imath})^2 + (j - \tilde{\jmath})^2 \right)$$

Dove $\tilde{\imath}$ e $\tilde{\jmath}$ sono le coordinate del centro di massa. Il problema con questa formula è la sua inefficienza quando usata come algoritmo. Vedremo adesso come renderla più efficiente per passaggi successivi.

$$\left(\sum_{i,j} A^i_j (i - \tilde{\imath})^2 \right) + \left(\sum_{i,j} A^i_j (j - \tilde{\jmath})^2 \right)$$

Concentriamoci adesso solo su uno dei due addendi. I passaggi saranno analoghi per l'altro.

$$\sum_i (i - \tilde{\imath})^2 \sum_j A^i_j$$

$$\sum_i (i - \tilde{\imath})^2 s^i(A)$$

$$\sum_i (i^2 - 2i\tilde{\imath} + \tilde{\imath}^2) s^i(A)$$

$$\sum_i i^2 s^i(A) - 2\tilde{\imath} \sum_i i s^i(A) + \tilde{\imath}^2 \sum_i s^i(A)$$

$$\sum_i i^2 s^i(A) - 2\tilde{\imath} \cdot \tilde{\imath} s(A) + \tilde{\imath}^2 s(A)$$

$$\sum_i i^2 s^i(A) - \tilde{\imath}^2 s(A)$$

$$\sum_i i^2 s^i(A) - \left(\frac{\sum_i i s^i(A)}{s(A)} \right)^2 s(A)$$

$$\sum_i i^2 s^i(A) - \frac{\left(\sum_i i s^i(A) \right)^2}{s(A)}$$

Ricordandoci dell'addendo che avevamo lasciato da parte, il calcolo diventa il seguente:

$$\sum_i i^2 s^i(A) + \sum_j j^2 s_j(A) - \frac{\left(\sum_i i s^i(A) \right)^2 + \left(\sum_j j s_j(A) \right)^2}{s(A)}$$

Nonostante il calcolo appaia più complicato di quello di partenza, è più semplice sia per un computer che per un umano. La parte complicata prima

stava nel trovare il quadrato di un numero reale, adesso dobbiamo fare il quadrato solamente di numeri interi.

Adesso però di quanto devo scalare l'immagine con minor momento di inerzia per fare in modo che abbia lo stesso momento di inerzia dell'altra? Per rispondere a questa domanda devo guardare la formula del momento di inerzia e rendermi conto che le i e le j (che in fisica corrispondono con le lunghezze) appaiano con un esponente quadratico. Quindi, se si scala l'immagine di un fattore r (ovvero si moltiplicano le i e le j per r e se ne sommano r volte tante), ottengo la stessa cosa che se si fosse moltiplicata l'espressione per r^4.

La conclusione è che devo scalare l'immagine di un fattore pari a:

$$r = \sqrt[4]{\frac{I_1}{I_2}}$$

Dove I_1 e I_2 sono i momenti di inerzia delle due immagini di partenza.

1.4.2 Indice di Jaccard

Una volta che le due lettere hanno le stesse dimensioni, possiamo usare l'indice di Jaccard per confrontarle.

L'indice di Jaccard è un indice che possiamo utilizzare per misurare la somiglianza fra due insiemi. Dati due insiemi A e B, l'indice di Jaccard si calcola come:

$$\frac{|A \cap B|}{|A \cup B|}$$

In questo caso l'insieme associato a ciascuna immagine comprende le posizioni dei pixel neri. Dunque l'indice rappresenta il rapporto fra il numero di pixel che sono neri in entrambe le immagini e il numero di pixel che sono neri in almeno una immagine.

Figura 1.9: In nero due lettere e a destra la rappresentazione del loro indice di Jaccard. In giallo la prima lettera, in blu la seconda e in verde la loro intersezione.

Figura 1.10: Le due immagini usare come test. Da notare che il bordo fine non fa parte dell'immagine.

Per avere una visione di come l'indice di Jaccard confronta due lettere possiamo guardare la figura 1.9. In tale figura si può vedere il confronto fra le lettere a sinistra. L'area occupata dalla sola lettera di figura di sinistra è colorata in giallo, quella occupata dalla sola lettera di figura di destra è colorata in blu e quella occupata da entrambe le lettere è verde. In questo caso l'indice di Jaccard risulta essere circa 0.53.

Se si utilizzano immagini scelte appositamente, come quelle di figura 1.10, si riesce a capire bene quali sono le differenze fra i vari metodi descritti nella sezione 1.4.1. In figura 1.11 si può apprezzare intuitivamente come funzionano i tre metodi. Si vede infatti che il metodo centrale allinea i centri delle immagini (in questo caso gli angoli dei quadrati), il metodo del baricentro allinea i baricentri (i centri dei quadrati), e il metodo inerziale, oltre ad allineare i baricentri, scala l'immagine più piccola in modo da avere lo stesso momento di inerzia della più grande (in questo esempio in modo che i quadrati abbiano le stesse dimensioni).

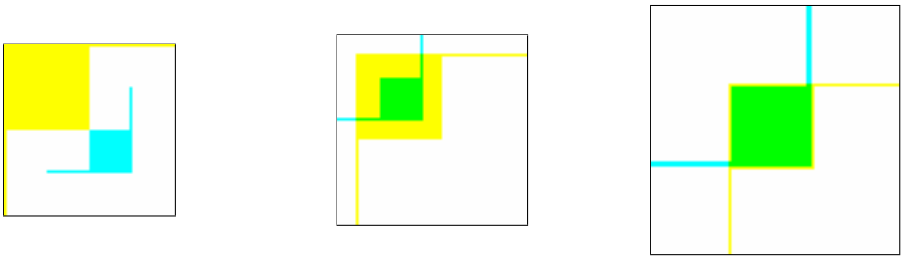

Figura 1.11: Le due immagini di test sovrapposte utilizzando le tre modalità. A sinistra il metodo centrale, nel mezzo il metodo del baricentro, e a destra il metodo inerziale.

Capitolo 2

Esperimenti

Per confrontare le varie modalità e avere una misura di quali siano le migliori non possiamo basarci solamente su analisi generali, ma dobbiamo fare degli esperimenti.

Per gli esperimenti verranno usate come confronti le lettere in figura 2.1.

Per valutare le prestazioni di ciascun algoritmo in maniera automatica viene usato un file .json che accompagna la scannerizzazione dell'immagine. In tale file ci sono indicate delle coordinate per ciascuna lettera di interesse. Per avere un unico punto associato a ciascuna lettera, si trova il punto medio fra quelli forniti, in modo da essere sicuri che questo sia in un punto interno

	d	*s*
dritta		
tonda		

Figura 2.1: Le lettere usate come confronto.

alla lettera.

Per trovare le lettere nel testo viene usata una funzione che riceve in ingresso un immagine sezionata secondo le modalità descritte alla sezione 1.2, l'immagine della lettera usata come confronto, l'indice di Jaccard da prendere come valore di soglia e un valore per indicare che modalità usare. Questa funzione dovrà dunque restituire una lista di rettangoli che sono i rettangoli contenenti le lettere che superano il test e i relativi valori di Jaccard. Ciascun rettangolo sarà rappresentato come una coppia di coordinate che si riferiscono a una coppia di vertici opposti.

2.1 Grafici precision recall

Una volta che abbiamo le rappresentazioni di tutti i rettangoli che l'algoritmo ritiene siano la lettera cercata, possiamo usare la lista di coordinate associate alle lettere corrette per quantificare gli errori dell'algoritmo. In particolare dobbiamo contare le volte in cui l'algoritmo ha creduto che una lettera fosse la stessa del riferimento sbagliandosi (falsi positivi) e le volte in cui ha creduto che una lettera non fosse la stessa del riferimento sbagliandosi (falsi negativi).

Idealmente vorremmo che sia i falsi positivi che i falsi negativi fossero 0. Tuttavia ciò è difficile da ottenere, in quanto alzando il valore di indice di Jaccard di soglia aumentano i falsi negativi, mentre abbassandolo aumentano i falsi positivi.

Nelle figure da 2.2 a 2.5 si possono visualizzare le affermazioni fatte precedentemente (i grafici relativi alla connettività a 8 sono stati omessi perché troppo simili ai corrispondenti per la connettività a 4). Nei grafici sull'asse delle ascisse c'è il recall (o sensibilità) e sull'asse delle ordinate la precision

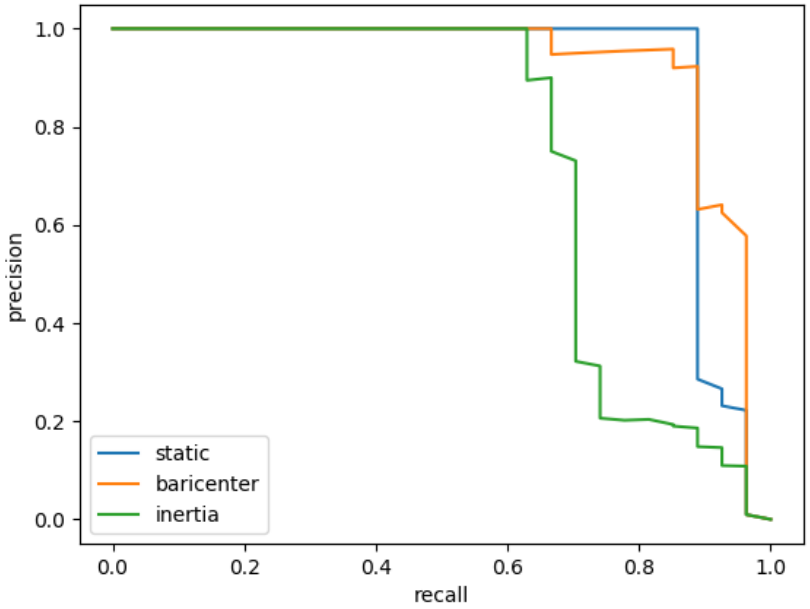

Figura 2.2: Curve di precision recall per la d dritta usando la connettività a 4.

(o predittività). Le definizioni di queste misure sono date in termini dei veri positivi V_+, i falsi positivi F_+, i veri negativi V_- e i falsi negativi F_-.

$$\text{recall} = \frac{V_+}{V_+ + F_-}$$

$$\text{precision} = \frac{V_+}{V_+ + F_+}$$

Ciò che si deduce da queste definizioni è che il recall è una misura degli errori intesi come falsi negativi, mentre la precision è una misura degli errori intesi come falsi positivi. Il caso ideale sarebbe avere precision e recall entrambi pari ad uno. Dai grafici è possibile confrontare i vari metodi per cercare di capire quale sia il migliore, ovvero quello che approssima meglio la curva ideale passante da $(1, 1)$.

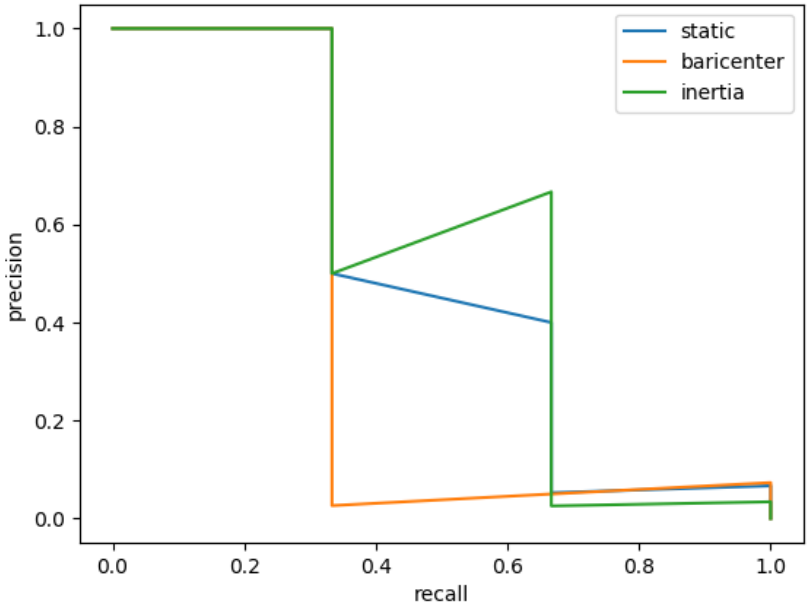

Figura 2.3: Curve di precision recall per la d tonda usando la connettività a 4.

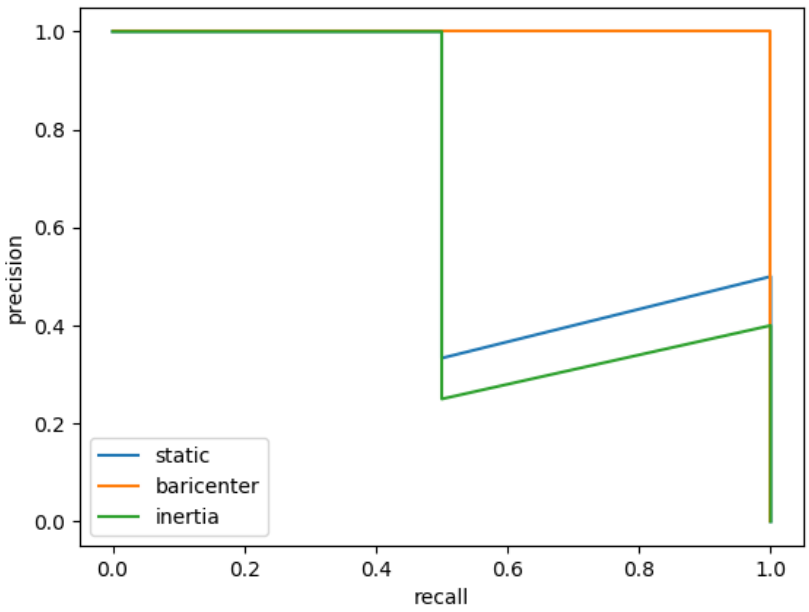

Figura 2.4: Curve di precision recall per la s dritta usando la connettività a 4.

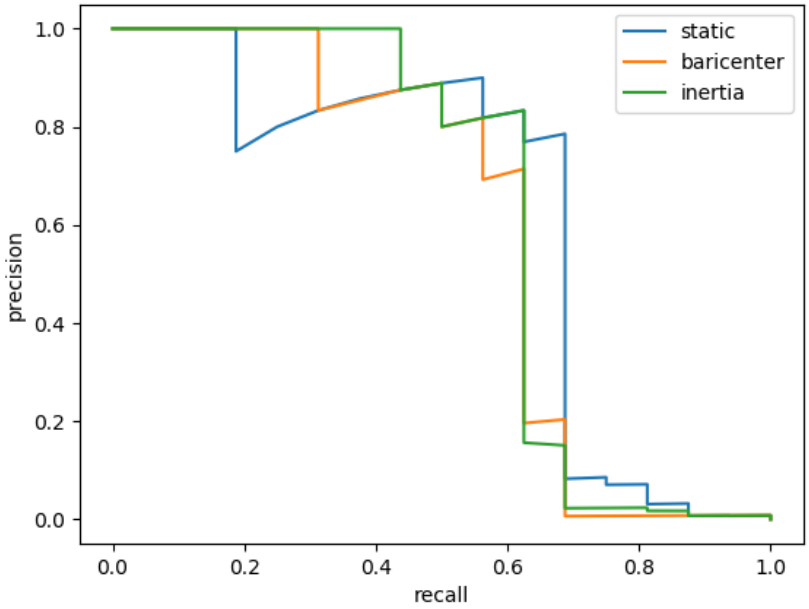

Figura 2.5: Curve di precision recall per la s tonda usando la connettività a 4.

2.2 Precision interpolata

I grafici possono essere semplificati facendone quella che viene chiamata interpolazione. In pratica ci si accorge che per alcuni tratti il grafico va "in salita", ovvero passa da un punto ad un altro con maggiore recall e precision. Questo significa che il secondo punto è migliore al primo secondo ogni misura. Detta in altri termini, usare il valore di soglia per l'indice di Jaccard relativo al secondo punto rispetto a quello del primo punto, comporta l'esclusione di grafemi incorretti, ma non di grafemi corretti e quindi è preferibile indiscutibilmente.

Di conseguenza possiamo fare in modo che il primo punto erediti la precision del secondo, in modo da passare dal grafico che indica "il miglior valore di precision ottenibile per un valore di recall pari a x" a quello che indica

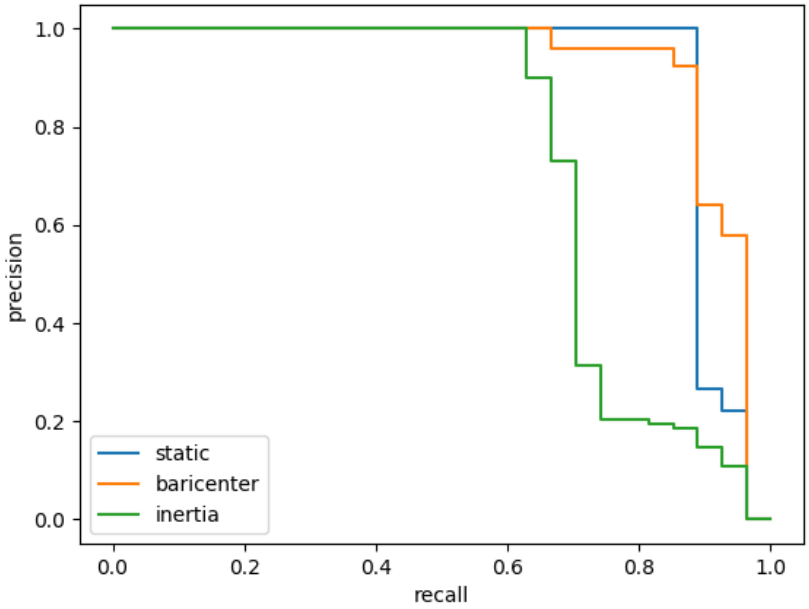

Figura 2.6: Curve di precision recall interpolate per la d dritta.

"il miglior valore di precision ottenibile per un valore di recall pari *o maggiore* di x". In questo modo si capisce bene che il grafico interpolato è più interessante.

Formalmente la precision nel grafico interpolato si trova come:

$$p_{interp}(r) = \max_{r' \geq r} p(r')$$

Dove p_{interp} è la precision interpolata, p la precision, e r un valore di recall.

Nelle figure da 2.6 a 2.9 viene raffigurata la versione interpolata dei grafici precision recall.

2.3 Media fra grafici

Usare quattro grafici per descrivere un unico metodo è eccessivo e dovremmo condensarli in un unico grafico per poter confrontare i metodi più

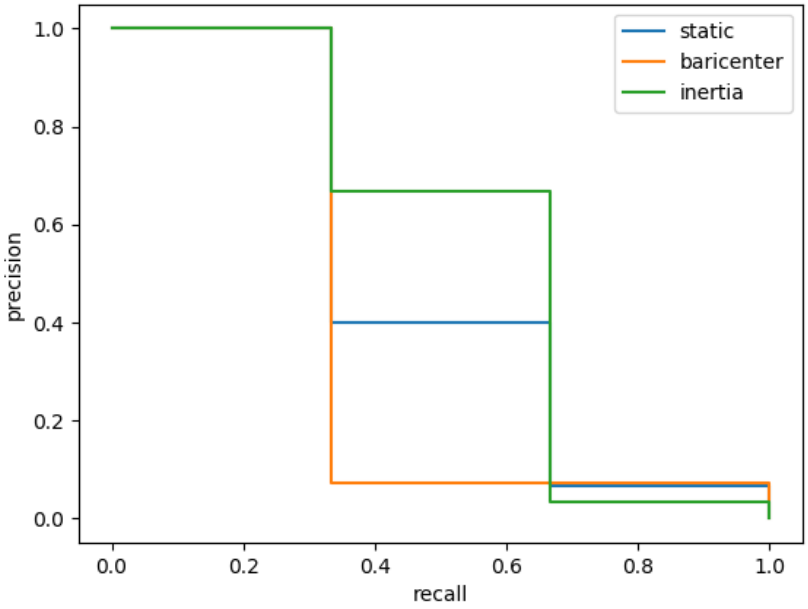

Figura 2.7: Curve di precision recall interpolate per la d tonda.

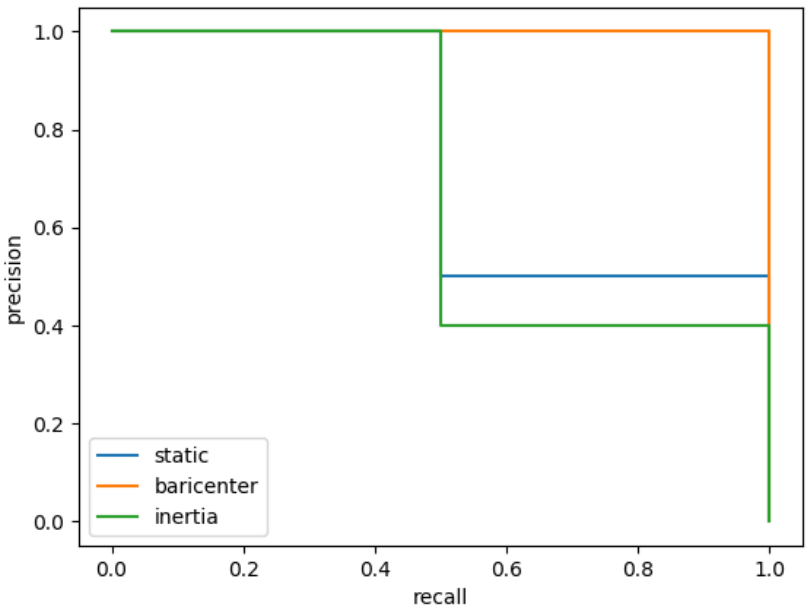

Figura 2.8: Curve di precision recall interpolate per la s dritta.

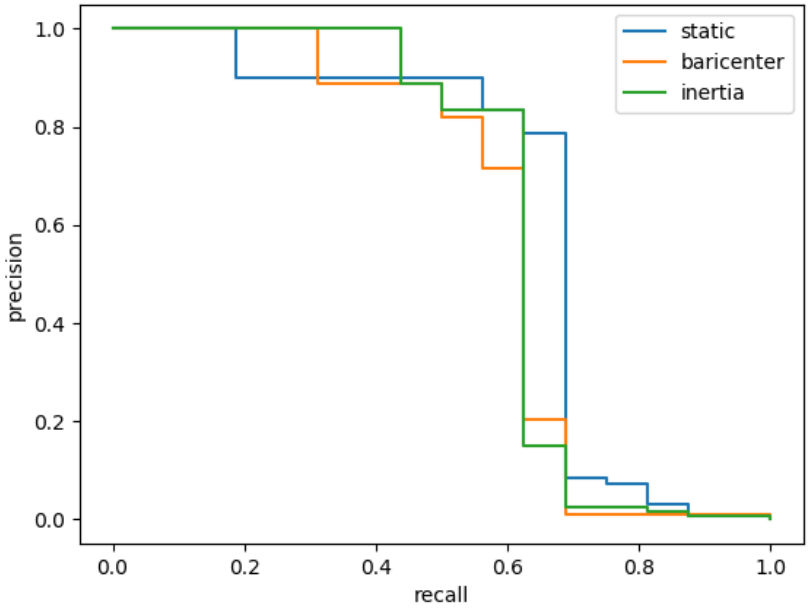

Figura 2.9: Curve di precision recall interpolate per la *s* tonda.

facilmente. Per fare ciò possiamo fare una media fra i grafici delle varie lettere ottenuti con lo stesso metodo per poi confrontare queste medie.

La media non deve essere una media aritmetica standard, poiché darebbe eguale peso a tutti i grafemi. Dobbiamo utilizzare una media pesata dove il peso di ciascun grafema è dato dalla sua frequenza nel testo. Utilizzando questo approccio, i grafici che otteniamo sono quelli nella figura 2.10. Come si evince dalle figure, e come è facilmente dimostrabile, la media fra grafici interpolati è a sua volta un grafico interpolato e quindi non è necessario farne l'interpolazione.

Da questi grafici possiamo notare che il metodo inerziale è il migliore per recall bassi, il metodo del baricentro è il migliore per recall alti, e il metodo statico è il migliore per recall intermedi.

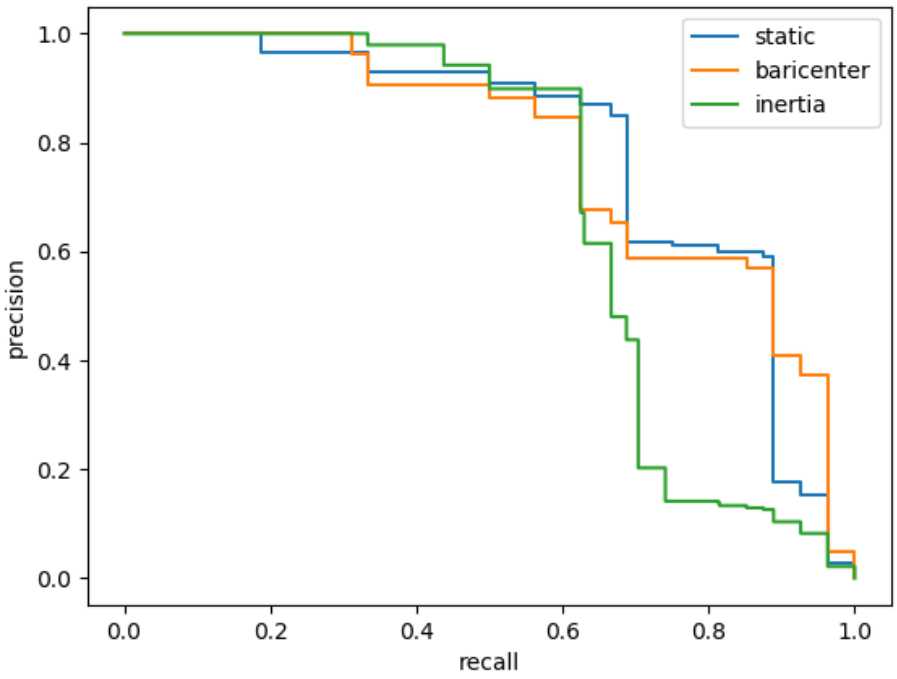

Figura 2.10: Curve di precision recall medie.

metodo	connettività 4	connettività 8
centrale	**0.785**	0.784
baricentro	0.782	0.781
inerzia	0.685	0.681

Tabella 2.1: Le aree sotto le curve precision recall per tutti i metodi testati. È stato evidenziato il valore massimo.

2.4 Area

Dai grafici, anche da quelli complessivi, è difficile capire quale metodo sia il migliore in generale. Quello che possiamo fare in questi casi è passare da un grafico ad un unico numero. Un modo per fare questo è calcolare l'area sottesa dal grafico. Quest'area sarà un valore compreso fra 0 e 1 e a metodi migliori corrisponderanno aree maggiori.

Le aree sono facilmente calcolate poiché sono semplicemente somme di aree di rettangoli. I loro valori sono riportati in tabella 2.1. Come si può notare i metodi corrispondono a aree simili, tutte vicine a 0.78, fatta eccezione per il metodo inerziale. Questo è probabilmente dovuto al fatto che le lettere sono tutte delle stesse dimensioni, mentre il metodo inerziale è efficacie quando le dimensioni sono variabili.

Capitolo 3

Interattività

Un modo semplice per aumentare l'accuratezza del programma è renderlo interattivo chiedendo aiuto all'utente per riconoscere le parole. Questo potrebbe sembrare insensato, visto che lo scopo del software è riconoscere i grafemi in maniera automatica, tuttavia non è così visto che il programma continuerebbe a fare la maggior parte del lavoro. Infatti, il programma fa tutto il processo descritto nei primi capitoli, fino a individuare le varie lettere e anche scartare quelle che sono molto diverse dal riferimento. L'unica cosa che viene chiesta all'utente è individuare i grafemi che corrispondono al riferimento scegliendo fra i pochi che rimangono.

3.1 Interfaccia interattiva

Per quanto riguarda l'interfacciamento con l'utente, questo avviene in modo molto semplice: si apre una finestra che chiede all'utente se la lettera mostrata è corretta che, quando l'utente ha risposto, si richiude per mostrarne un'altra. In questo modo l'utente deve fare una sola azione (premere un tasto)

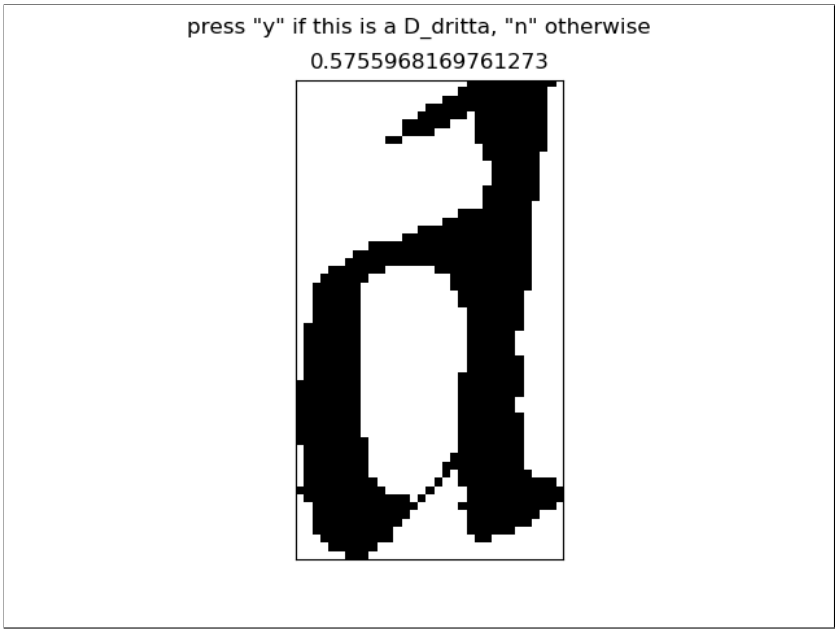

Figura 3.1: L'interfaccia grafica sviluppata. In alto le indicazioni e l'indice di Jaccard nella seconda riga.

per ogni grafema mostrato, quindi viene ridotta al minimo la parte di lavoro non automatizzata.

L'interfaccia che vede l'utente è quella di figura 3.1. Nella prima riga c'è il comando in cui si dice di premere il tasto y ("yes") se la lettera è quella che si sta cercando o il tasto n ("no") in caso contrario. Viene anche mostrato l'indice di Jaccard relativo alla lettera per avere un'informazione aggiuntiva.

3.2 Interattività limitata

Nel caso di documenti molto lunghi, comprendenti dunque molte pagine, chiedere conferma all'utente per ogni lettera, significherebbe chiedergli troppo lavoro. Tuttavia ciò non significa che l'interattività non sia una buona idea.

La soluzione consiste nell'integrare una parte interattiva con una parte completamente automatica. In pratica, durante la parte interattiva si raccol-

gono nuovi riferimenti che verranno poi utilizzati nella parte completamente automatica. Questo significa che nel primo round verranno selezionate soltanto alcune lettere. Per scegliere le lettere da usare nel primo round è necessario usare un generatore di numeri casuali, altrimenti si possono avere effetti indesiderati (se, ad esempio, analizziamo solamente una pagina potrebbe succedere che la grafia di pagine successive sia diversa da quella della pagina selezionata).

3.2.1 Tipi di media

Una volta che abbiamo l'insieme di riferimenti trovati in maniera casuale e interattiva, possiamo calcolare, per ogni nuova lettera, gli indici di Jaccard relativi a ciascun riferimento. Adesso però dobbiamo trovare un modo per passare da tanti valori ad uno solo indicativo di tutti.

Ciò che stiamo cercando è una funzione che faccia da media. Potremmo pensare di usare la media aritmetica, ma questa non ha certe proprietà che stiamo cercando. Infatti vogliamo che la funzione restituisca 1 quando almeno uno degli elementi è 1. Questo perché il fatto che uno dei valori sia 1 equivale a dire che la lettera che stiamo testando è identica ad un riferimento e perciò è corretta sicuramente.

A questo punto potremmo pensare di usare come media la funzione *max*, che restituisce il massimo dei valori in ingresso. Questa funzione ha infatti la proprietà di cui abbiamo parlato prima: se 1 è fra i valori in ingresso, allora *max* restituirà sicuramente 1 (ricordiamo che i valori di ingresso sono indici di Jaccard e percò sono sempre compresi fra 0 e 1 inclusi). Tuttavia, neanche questa funzione è adatta, poiché non tiene conto dei valori più bassi. Questo è un problema perché passare da una lista di valori $(\frac{1}{2})$ a $(\frac{1}{2}, \frac{1}{4})$ deve far diminuire l'indice risultante.

Dopo queste considerazioni arriviamo alla seguente formula.

$$1 - \prod_{0 \leq i < n} (1 - x_i)$$

In cui n è il numero di valori in ingresso. Con questa funzione tutti i valori contribuiscono al risultato finale a meno che uno dei valori sia 1, in qual caso il risultato è 1. Ancora però non è perfetta come funzione, infatti se prendo $A = (\frac{1}{2}, \frac{1}{2})$, il risultato non è $\frac{1}{2}$ come sarebbe ovvio aspettarsi, ma $\frac{3}{4}$.

Dopo un ultimo ritocco alla formula, si arriva alla forma seguente.

$$1 - \sqrt[n]{\prod_{0 \leq i < n} (1 - x_i)}$$

Questa ha tutte le proprietà della funzione precedente e in più, se tutti i valori sono uguali, restituisce il valore in ingresso.

3.3 Confronto con modalità non interattiva

Per verificare se l'introduzione di interattività ha aiutato il programma dobbiamo confrontarne i risultati con quelli che ottenevamo senza interattività. Nelle figure 3.2, 3.3 e 3.4 sono mostrati i grafici precision recall per le varie modalità.

Come si nota nelle figure, le curve relative alla non interattività sono diverse rispetto a quelle del capitolo precedente. Questo perché sono state utilizzate pagine diverse per enfatizzare il punto forte dell'interattività: è efficace con pagine scritte con calligrafie diverse.

È evidente come il metodo inerziale sia il peggiore dei tre in questa modalità. Infatti anche in questo caso le lettere sono tutte della stessa grandezza più o meno. Si faccia caso, però, al fatto che il metodo inerziale è il migliore nella modalità non interattiva. Questo è dovuto al fatto che, come avevamo

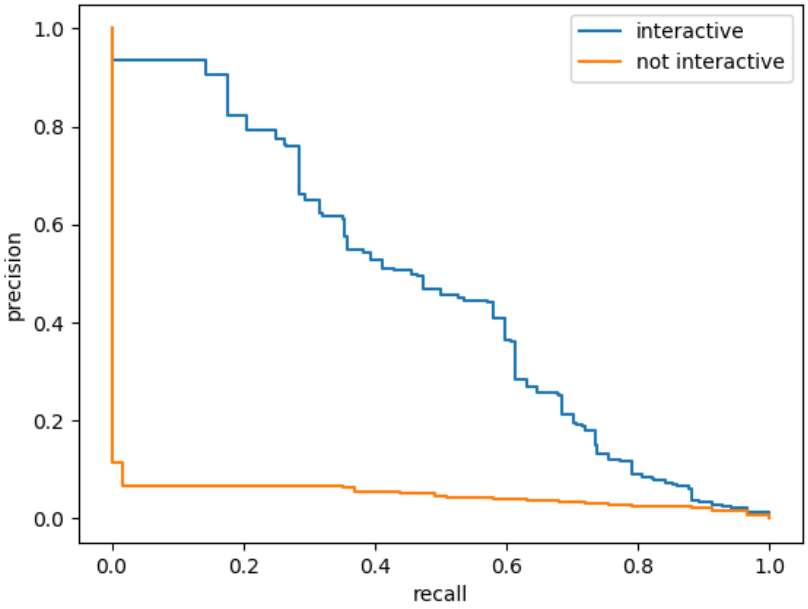

Figura 3.2: Curve precision recall con e senza interattività, metodo centrale.

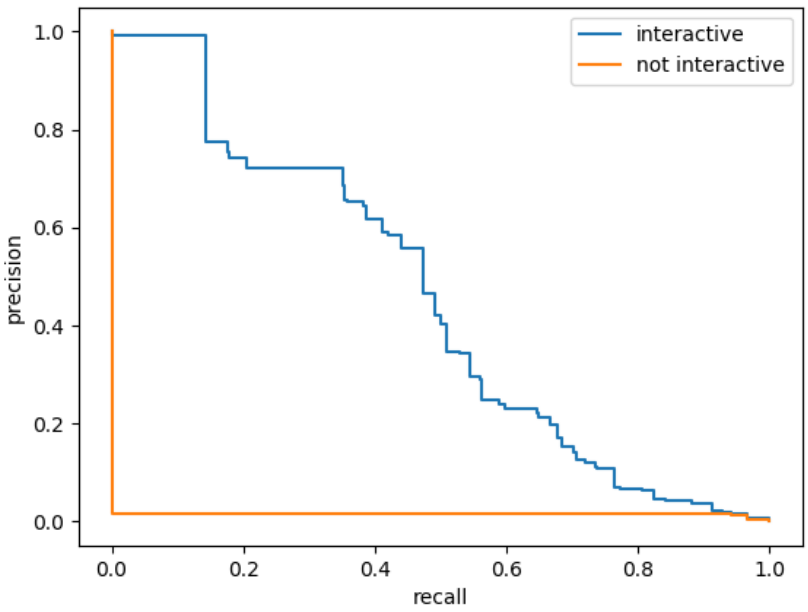

Figura 3.3: Curve precision recall con e senza interattività, metodo del baricentro.

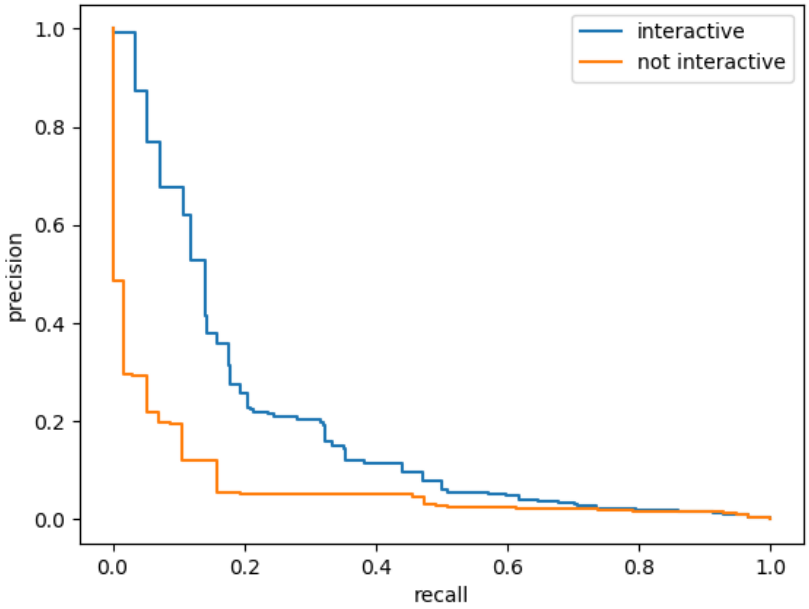

Figura 3.4: Curve precision recall con e senza interattività, metodo inerziale.

predetto nella sezione 1.4.2, questo metodo è adatto ai casi in cui le pagine hanno stili differenti e in particolare hanno lettere di dimensione variabile.

Come prima, mostriamo anche le misure delle aree sotto le curve di precision recall nella tabella 3.1. Va ricordato che questi valori non sono deterministici come quelli del capitolo 2, ma dipendono dalla scelta casuale delle lettere estratte e usate per l'interattività.

connettività	modalità	interattivo	non interattivo
4	centro	0.463	0.047
	baricentro	0.448	0.017
	inerzia	0.188	0.063
8	centro	0.449	0.046
	baricentro	0.391	0.017
	inerzia	0.193	0.063

Tabella 3.1: Le aree sotto le curve precision recall per tutti i metodi interattivi testati e il confronto rispetto agli stessi metodi senza interattività.

Capitolo 4

Conclusioni

Quello che abbiamo fatto è stato, a partire da digitalizzazioni di documenti manoscritti medievali, trovare un metodo per l'estrazione di alcuni grafemi che ci interessavano.

Prima di tutto abbiamo binarizzato le immagini per rendere più semplici le operazioni successive. Dopodiché abbiamo utilizzato metodi basati su proiezioni orizzontali e verticali per suddividere il documento nelle righe e nei caratteri che lo compongono.

Una volta isolate le varie immagini dei caratteri e estratti da essi i grafemi mediante lo studio delle componenti connesse, siamo passati a confrontarle con dei modelli estratti manualmente in precedenza per trovare quelle che rappresentano ciascuno dei grafemi che ci interessano.

Il confronto avviene mediante vari passaggi successivi. Inizialmente si trasformano le immagini per renderle tutte delle stesse dimensioni. Questo avviene secondo varie modalità che verranno confrontate fra di loro. Infine si confrontano le immagini tramite l'indice di Jaccard.

Per confrontare la bontà delle varie modalità si usano degli scan già analizzati manualmente. I risultati degli esperimenti sono riportati nel capitolo

2, da cui si vede che tutte le modalità si comportano in maniera molto simile, con piccole differenze.

Come ultima cosa si è usata un'interfaccia interattiva per rendere il programma migliore in situazioni in cui non basta un unico modello per analizzare tutto il documento. Questo è stato descritto nel capitolo 3, in cui si può vedere il miglioramento considerevole rispetto alla modalità non interattiva.

In conclusione, il programma risulta utile per la ricerca dei grafemi che ci interessavano e si rivela anche versatile tramite l'uso della modalità interattiva, nonostante l'apparente semplicità delle tecniche utilizzate.

Bibliografia

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[3] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[4] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[5] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt,

and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.