Metodi di Ingegneria della Conoscenza applicati alle homepage delle scuole superiori italiane

Corso didattico

- Ingegneria della Conoscenza [063507], Facoltà di "Informatica"
- A.A. 2022/23

Gruppo di lavoro

- Vincenzo Di Bisceglie [745751] v.dibisceglie3@studenti.uniba.it

Repository

- https://github.com/vodibe/icon-74571

Sommario

Introduzione

Idea del progetto

L'idea di fondo da cui si è partiti per lo sviluppo di questo progetto è l'applicazione di alcuni metodi di Ingegneria della Conoscenza su un dominio di interesse, in questo caso l'usabilità di una pagina web. Questo richiede che prima si vada a circoscrivere un ambito di riferimento, che nel nostro caso, è l'insieme delle Homepage delle scuole superiori pubbliche italiane (aggiornate a settembre 2023).

Metriche di usabilità già esistenti

Le metriche rilevanti proposte in letteratura e che potrebbero essere applicate nel contesto di questo progetto sono le Euristiche di Nielsen [1] le WCAG 2.1 [2] per le quali però gli strumenti software ad essi correlati (qui elencati) non sono adatti alla natura di questo progetto perché non esprimono una valutazione numerica, ma analizzano il codice sorgente della pagina e danno consigli per rimediare le linee guida non rispettate. Altri strumenti controllano condizioni di accessibilità da parte di utenti con handicap (ad es. verificano che la palette di colori sia accessibile, controllano l'interazione con hardware ausiliari, …)

Altre metriche rilevanti (SUS Score [3]) non sono state prese in considerazione perché richiedono un campione di persone alle quali sottoporre un questionario.

Metrica di usabilità adottata per questo progetto

Ai fini del progetto assumeremo che questa nuova metrica di usabilità corrisponde a un voto assegnato da una persona che non ha mai interagito con la Homepage prima d'ora, tenendo conto di quanto l'interfaccia sia ordinata e funzionale. **Questa metrica verrà vista come un qualcosa di condiviso dai visitatori (concetto oggettivo), e non come una valutazione soggettiva che uno specifico utente dà alla pagina**. Approfondiremo questa metrica nelle sezioni successive.

Elenco argomenti di interesse

Fasi del progetto e per ciascuna di esse gli argomenti coinvolti:

1. **Costruzione del ground truth**.
   Poiché all'inizio non disponiamo di una valutazione per tutte le Homepage, ci immedesimiamo in un visitatore della pagina, ne osserviamo gli aspetti grafici e funzionali (in altre parole osserviamo il valore di alcune feature iniziali), e diamo una valutazione. Gli step seguiti sono:

   1.1. Preprocessing del dataset delle scuole.

   1.2. Raccolta dei dati in un dataset rappresentante il ground truth.

2. Emulazione del ground truth.
   La fase 1 prevede un'osservazione diretta della grafica, e ciò ovviamente non può essere automatizzato, ma deve essere valutato con criterio. Pertanto in questa fase riproduciamo il ground truth utilizzando strumenti che si prestano meglio all'elaborazione e apprendimento automatico. Gli step seguiti sono:

   2.1. Osservazione di caratteristiche della pagina ottenibili in modo automatico per ciascun sito, mediante **Rappresentazione dello spazio di ricerca tramite grafo**.

   2.2. Costruzione e valutazione di **Modelli di apprendimento supervisionato** che, a partire dalle feature per ciascun sito (individuate al punto 2.1) simulano la sua valutazione.

3. Deduzione di informazioni utili.

   3.1. Costruzione di una KB e **ragionamento relazionale sfruttando anche il Web Semantico**.

   3.2. **Costruzione di un modello probabilistico** e suo impiego per task di inferenza probabilistica.

Costruzione del ground truth

Sommario

Si è ipotizzato che in generale un visitatore quando osserva la pagina, può assegnare il grado di usabilità con una scala [1, 5].

- [1, 2): **SITO ESTREMAMENTE CONFUSO**
  Non esiste un menu; la disposizione di tutti gli elementi è disordinata, per cui è difficile individuare le sezioni che l'utente vuole visitare.
- [2, 3): **SITO CONFUSO** **ES: HTTPS://WWW.GALILEIFERRARI.IT/**
  Esiste un menu; la disposizione di quasi tutti gli elementi della pagina è disordinata e la pagina dà l'impressione di essere troppo lunga.
- [3, 4): **SITO ACCETTABILE** **ES: HTTPS://WWW.ISII.IT/**
  Esiste un menu che reindirizza il visitatore a gran parte delle sezioni di suo interesse; la pagina però contiene un discreto numero di elementi non raggruppati e quindi confusionari.
- [4, 5]: **SITO ORDINATO** **ES: HTTPS://WWW.EINSTEINRIMINI.EDU.IT/**
  Sito accettabile e che inoltre contiene pochi o nessun elemento non raggruppato.

Decisioni di progetto

Si suppone che l'utente vada ad assegnare un valore di usabilità alla pagina ragionando su alcuni fattori. Per comodità, è utile raccogliere i fattori di decisione e la valutazione finale in un dataset che chiamiamo `ds2_gt`.

La prima cosa che consideriamo vedendo una pagina web scolastica può essere la presenza di un menu, per cui si introduce la feature discreta `page_menu_or` che ne descrive l'orientamento.

Poi viene introdotto un secondo fattore, il più rilevante, dovuto al fatto che nella quasi totalità dei siti scolastici ritroviamo il "trend" di inserire dei banner che linkano a una sezione del sito. Spesso, tali banner sono difficili da leggere e posti sulla pagina in modo disordinato, cioè non raggruppati in un menu o in una sezione specifica della pagina. Per generalizzare (includendo qualsiasi contenuto multimediale, e quindi anche video) introduciamo la feature `page_ungrouped_multim`.

Figura 1. Esempi di banner.

Per ultimo, c'è la feature `page_template`, utile a fornire un contesto in cui "inquadrare" la feature `page_ungrouped_multim`. Questo accade in quanto possono esistere più pagine che, seppur hanno lo stesso numero di elementi multimediali non raggruppati, risultano in una valutazione diversa perché basate appunto su template diversi.

Potremmo ipotizzare che la valutazione possa dipendere anche da quanto sia lunga la pagina, tuttavia un visitatore non viene mai a conoscenza dell'altezza precisa (in pixel). Pertanto non è stata considerata.

Fattore di decisione	*Descrizione*	*Dominio*	*u.m.*
`page_template`	Template adottato (vedi Figura 2) 1,...,8=template ID 9=non segue un template	$\{1,2,\ldots,9\}$	
`page_menu_or`	Orientamento menu. 0=non esiste 1=solo orizzontale 2=solo verticale 3=orizzontale e verticale	$\{0,1,2,3\}$	
`page_ungrouped_multim`	Elementi grafici non raggruppati.	$\mathbb{N}$	

A quale pagina web sono associati questi fattori?

Feature PK	*Descrizione*	*Dominio*	*u.m.*
`school_id`	Codice della scuola.	Stringhe	
`page_url`	URL della pagina.	Stringhe	

La valutazione è la seguente:

Feature	*Descrizione*	*Dominio*	*u.m.*
`metric`	Valutazione di usabilità della pagina.	$[1,\ldots,5]$	

In Figura 2 (di seguito) sono elencati tutti i template ad oggi impiegati dai siti scolastici italiani:

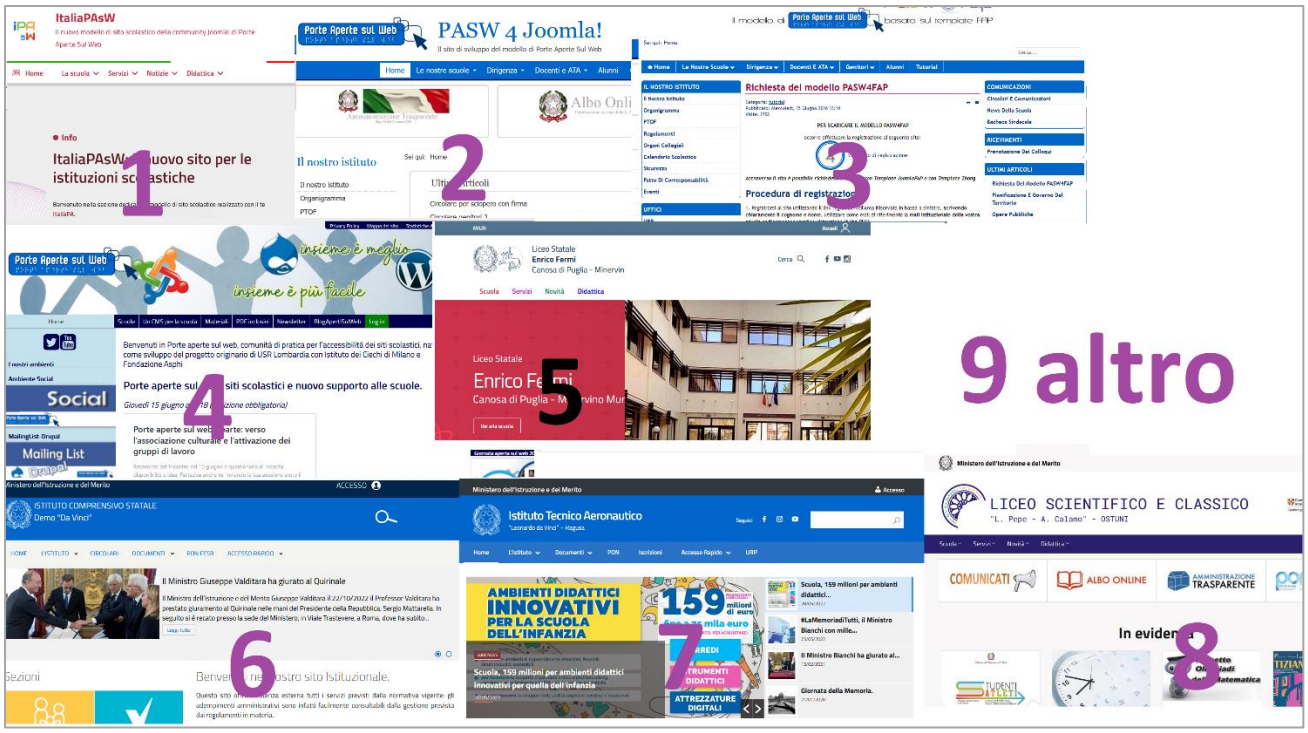


Figura 2.

1. https://paswjoomla.net/Jipa4school/
2. http://paswjoomla.net/pasw/
3. http://paswjoomla.net/jfap/
4. https://www.porteapertesulweb.it/
5. https://italia.github.io/design-scuole-pagine-statiche/scuole-la-scuola.html
6. https://italiajoo.demoargoweb.com/
7. https://italiawp.demoargoweb.com/
8. https://www.liceopepecalamo.edu.it/

Preprocessing del dataset delle scuole

Codice: **`/agent/preproc/dataset_creator.py`**

Una volta introdotta la metrica, è necessario parlare della fase di preprocessing del dataset iniziale. Il catalogo offerto dal MIUR raggruppa le informazioni su tutte le scuole (elementari, medie e superiori) pubbliche. Durante la fase di preprocessing si vanno a creare, in ordine, i seguenti DS:

1. **`ds1`**: https://dati.istruzione.it/opendata/opendata/catalogo/elements1/?area=Scuole
   Le feature di questo DS sono descritte in questa pagina web.
2. **`ds1_clean`** ottenuto inserendo solo le scuole superiori ed effettuando un preprocessing sull'URL che consiste nel vedere se il sito corrente è rintracciabile con una semplice richiesta HTTP. Se non lo è, ed inoltre il sito ha un TLD diverso da **`.edu.it`**, si sostituisce il TLD corrente con **`.edu.it`**. Se un sito non è rintracciabile neanche dopo aver effettuato la sostituzione, lo si esclude dal DS.
3. **`ds1_clean_unique`** ottenuto rimuovendo i siti duplicati. Operazione necessaria in quanto se un plesso scolastico offre più corsi di studio (ad es. istituto tecnico e istituto professionale) e ha un singolo sito web, ciascun corso ricopre una riga nel **`ds1`**.
4. **`ds2_gt`** ottenuto richiedendo i fattori di decisione e la valutazione per ciascun sito presente in **`ds1_clean_unique`**.
5. **`ds3_gt`** ottenuto inserendo tutte le features necessarie per addestrare un modello di apprendimento. Verranno descritte nel capitolo dedicato.
6. **`ds3_gt_final`**. E' possibile che il **`ds3_gt`** contenga qualche URL raggiungibile ma non valido, dovuto al fatto che il sito è in manutenzione, ha subito un cambio dominio o che faccia riferimento a una scuola superiore erroneamente catalogata nel **`ds1`** (ad esempio esclusivamente serale). Pertanto, queste righe vengono rimosse in questo nuovo DS.

In sintesi (Figura 3):

- Siti di scuole superiori ma che non sono raggiungibili, oppure siti di altre scuole (medie, convitti, …)
- (accanto a *unique*) Siti duplicati.
- (accanto a *final*) Siti raggiungibili ma non validi.
- Siti delle scuole superiori raggiungibili e validi.

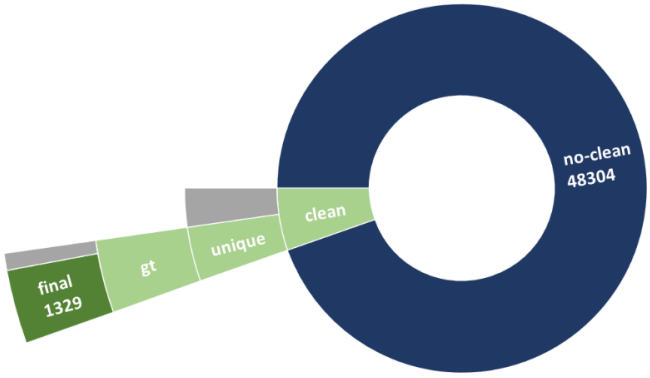


Figura 3. Proporzione del numero di elementi dei vari DS.

Rappresentazione dello spazio di ricerca con grafo e ricerca soluzioni

Sommario

Per eseguire il task di apprendimento del ground truth, dobbiamo individuare alcune features che possano essere osservate automaticamente a partire dalla pagina web. Per cui, notando che una pagina web equivale a un DOM, possiamo usare il concetto di rappresentazione dello spazio di ricerca tramite grafo.

Strumenti utilizzati: modello NaiveDOM

Codice: `/agent/ndom/NaiveDOM.py`

In questo progetto è stato introdotto il concetto di NaiveDOM (NDOM) che è un modello DOM semplificato di una pagina web ottenuto dal parsing del codice sorgente HTML. I suoi dettagli teorici sono presentati nella prossima sezione.

Struttura del NDOM

Un NDOM è un grafo diretto e pesato, avente struttura ad albero. E' tale per cui:

- Ha numero finito di nodi ed è aciclico (diretta conseguenza del fatto che è un DOM semplificato)
- Ciascun nodo è un elemento della pagina, e quindi è identificato univocamente dal suo XPath [4].
  A ciascun nodo sono associati una label (per fini di rappresentazione grafica) e le sue coordinate (x, y) all'interno della pagina renderizzata.
- Il **nodo radice** è l'XPath del tag **`<body>`**.
- I **nodi interni** sono gli XPath dei tag che contengono potenzialmente, tra i loro discendenti, un testo leggibile. Ad es. **`<body>`**, **`<header>`**, **`<section>`**, **`<nav>`** ecc... Sono esclusi i tag **`<div>`**, visto che sono assai frequenti e non semplificano (ma complicano) la struttura del NDOM.
- I **nodi foglia** possono essere di tre tipi:
  - XPath dei tag che non contengono un testo leggibile, ad es. **`<img>`**, ecc...
  - XPath dei tag che contengono sicuramente un testo leggibile, ad es. **`<a>`**, **`<h1>`**, ecc...
  - Il testo leggibile, a patto che abbia una lunghezza breve.
- Come un qualsiasi albero, ha una sua **altezza**, cioè un numero indicante la massima profondità di un nodo.
- Per quanto riguarda gli **archi** del NDOM e il loro costo, è necessario prima osservare direttamente un esempio di NDOM costruito per una pagina. Si veda la prossima sezione.

I dettagli implementativi di questo modello sono descritti nella sezione *Decisioni di progetto*.

Calcolo del costo degli archi

Visualizziamo un sito scolastico, e rappresentiamo in forma stilizzata il suo NDOM.

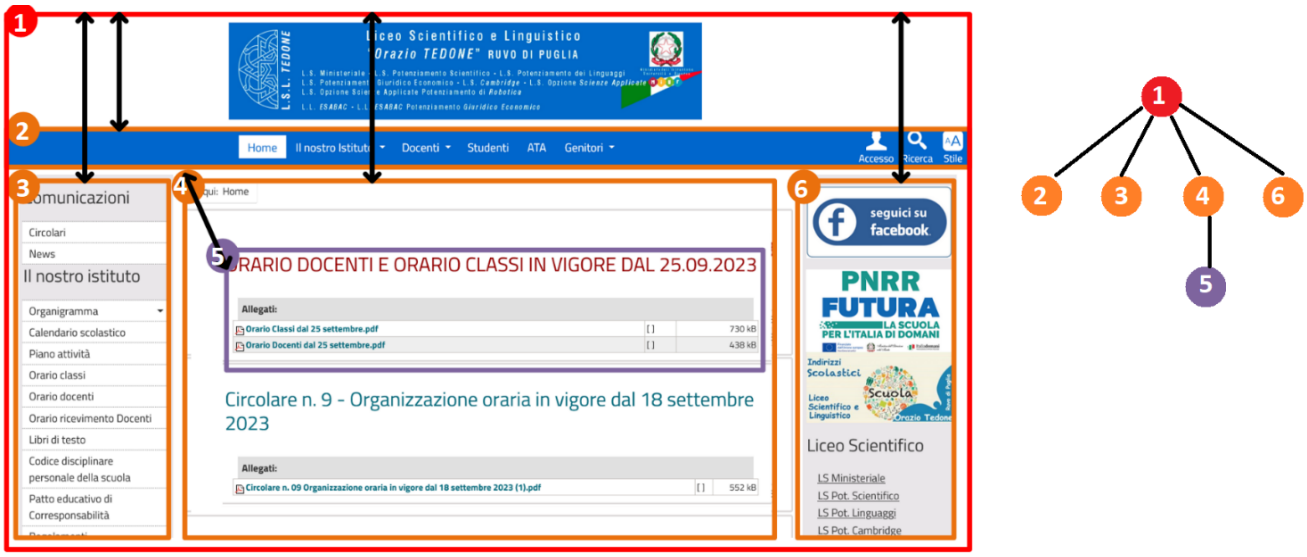


Figura 4. https://www.liceotedone.edu.it/ . Rappresentazione stilizzata del NDOM.

Osservando lo screenshot di questo sito web (Figura 3), notiamo che il nodo radice **<body>** ha ovviamente coordinate (0,0). I rettangoli arancioni indicano elementi della pagina innestati all'interno del tag **<body>**. Solo per questi elementi (figli diretti della radice del NDOM), la distanza tra padre e figlio è puramente verticale: questo è ovvio perché l'occhio umano inizia osservando dal basso verso l'alto. In tutti gli altri casi si provvede a calcolare la distanza euclidea.

Il **costo dell'arco tra padre-figlio** è una funzione della distanza padre-figlio (di seguito chiamata x), ed è calcolata in `_calc_arc_cost`. Essenzialmente si riconduce alla seguente funzione:

$$c(distanza) = \frac{distanza}{diagonale(1600{,}900)} * e^{1.3} \qquad x \geq 0$$

La funzione descritta di calcolo del costo padre-figlio ha il seguente comportamento. La linea verde chiaro fa riferimento agli schermi con risoluzione 1600x900, quella verde scuro agli schermi 1020x1080. Man mano che aumenta la distanza in pixel tra un elemento, aumenta il costo in termini di usabilità. A parità di distanza, il costo (su schermi con risoluzione minore) aumenta.

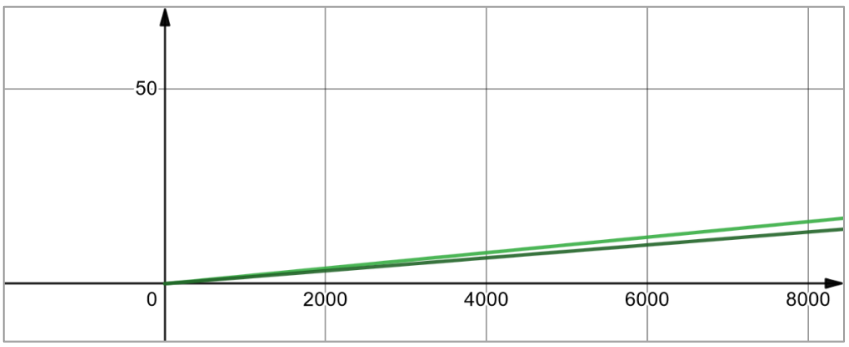


Figura 5. Grafico funzione $c(distanza)$ di costo dell'arco padre-figlio.
https://www.desmos.com/calculator/aaqy3tao8g

Calcolo di un task con algoritmo di ricerca

Vedere: `/agent/ndom/NaiveDOM.py`
`/agent/ndom/NaiveDOMSearcher.py`

La costruzione del NDOM di una pagina web richiede un'istanza di un browser automatizzato che disponga di un interprete JS. Grazie ad esso, è possibile ricavare altre due feature inerenti ad essa: **`page_width`** e **`page_height`**. Al termine, siamo in grado di ingegnerizzare 13 nuove feature:

Feature	*Descrizione*	*Dominio*	*u.m.*
`page_load_time_ms`	Tempo di caricamento della pagina.	$\mathbb{N}$	ms
`page_width`	Larghezza della pagina.	$\mathbb{N}$	px
`page_height`	Altezza della pagina.	$\mathbb{N}$	px
`NDOM_nodes`	Numero di nodi del NDOM associato alla pagina.	$\mathbb{N}$	
`NDOM_height`	Altezza del NDOM associato alla pagina.	$\mathbb{N}$	
`task1`	Costo in termini di usabilità per svolgere il task #1.	$\mathbb{R}$	
	...		
`task8`	Costo in termini di usabilità per svolgere il task #8.	$\mathbb{R}$	

Come si calcola il valore della feature `taskx`?

Innanzitutto, un Task è una sezione che l'utente è interessato a raggiungere e che, se individuata, in un certo senso rispecchia parte di usabilità della pagina. Un Task contiene un Task ID e delle Task Keywords, cioè una lista di stringhe tali per cui, se l'utente ne individua una all'interno della pagina, porta a termine il suddetto Task. Il dizionario dei Task è mostrato di seguito, e raccoglie alcune sezioni tipiche di un sito scolastico.

```
TASKS_DEFAULT = {
    "task1": ["circolari", "comunicazioni", "circolare"],
    "task2": ["organigramma", "organizzazione", "schema organizzativo", "persone"],
    "task3": ["notizie", "news", "eventi"],
    "task4": ["progetti", "progetto", "projects"],
    "task5": ["regolamento", "regolamenti", "regolamentazione"],
    "task6": ["amministrazione trasparente", "ammin. trasparente"],
    "task7": ["registro"],
    "task8": ["indirizzo", "i luoghi", "dove siamo", "contatti"],
}
```

Figura 6.

A questo punto, l'algoritmo di ricerca proposto (chiamato nel codice come **`NaiveDOMSearcher`**) cerca di emulare il comportamento dell'occhio umano, e ciò è rappresentato dall'immagine in Figura 7: un percorso per il quale l'ultimo nodo ha profondità 0 o 1 (vale a dire, il nodo radice e tutti i percorsi dal nodo radice ai suoi figli diretti) viene aggiunto a una coda con priorità, in cui il percorso a costo minore sarà il primo ad essere esaminato. Questo è ovvio perché una persona passa ad esaminare prima le voci del menu principale rispetto alle voci del footer (che si trovano a fine pagina). Successivamente, gli alberi radicati nei figli diretti della radice vengono esaminati in modalità DFS.

Di seguito è illustrato il suo funzionamento.

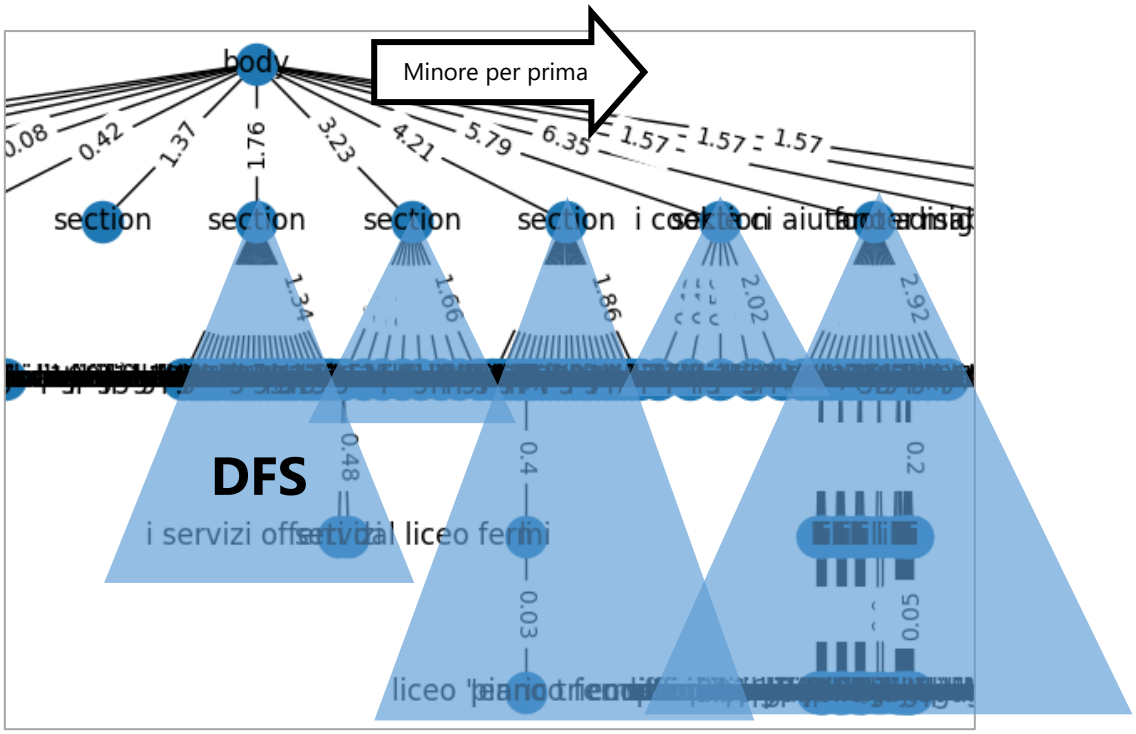


Figura 7. Metodo ***`plot()`*** *chiamato sul NDOM di https://www.liceofermicanosa.edu.it/ e illustrazione del funzionamento dell'algoritmo di ricerca.*

Questo algoritmo di ricerca gode delle seguenti proprietà:

- E' completo, cioè certo di trovare un nodo obiettivo se esso esiste.
- Non va in loop (diretta conseguenza della struttura del NDOM)
- Come l'algoritmo DFS, ha complessità di spazio $O(b * h)$ ove b è il branching factor e h è la profondità del nodo goal; complessità di tempo $O(|Nodi|) = O(b^h)$.
- E' ottimale?

Se esiste un percorso dal nodo radice a un nodo obiettivo per il task, chiamiamo con c il costo del percorso (cioè la somma di tutti i costi degli archi) e applichiamo la seguente funzione che aggiunge alcune penalità.

$$costoTotaleTask(c) = c + Penalità(Numero\ percorsi\ già\ espansi) + Penalità(Numero\ nodi\ del\ percorso)$$

Ad esempio:

$$costoTotaleTask(c) = c + \left(\frac{Num.\ percorsi\ già\ espansi}{280} - 0.2\right) + (Num.\ nodi\ del\ percorso * 0.15)$$

La prima penalità dipende dal numero di percorsi già esaminati prima di individuare quello dalla radice al nodo obiettivo. La seconda è dovuta al fatto che, se ipotizziamo che il nodo A ha come figlio il nodo B, il passare dall'interagire con il nodo A all'interagire con il nodo B richiede raramente (ma comunque non è impossibile) un'operazione scomoda da fare, come un click, l'attesa di un'animazione ecc. …

Idealmente, un sito con costi molto bassi per i task è tale per cui tutte le sezioni utili non sono sparse nella pagina ma sono elencate chiaramente o in menu a tendina.

Se non esiste almeno un nodo obiettivo per un Task, si assegna al Task un costo di default. Questa casistica avviene quando nella pagina non c'è una stringa visibile che soddisfa il Task. Le cause sono:

- il designer del sito non prevede l'inserimento di una sezione correlata al Task (grave).
- il nodo obiettivo non è una stringa visibile, ma un'immagine (quasi sempre un banner) (comprensibile).

In questo progetto NON sono state implementate tecniche per gestire la seconda causa (ad es. tecniche OCR), per cui il costo di default è di media entità, risultante dal compromesso tra la prima e la seconda causa.

$$costoDefaultTask(|NDOM|) = \ 6.5 + floor\left(\frac{|NDOM|}{500}\right) * 0.5$$

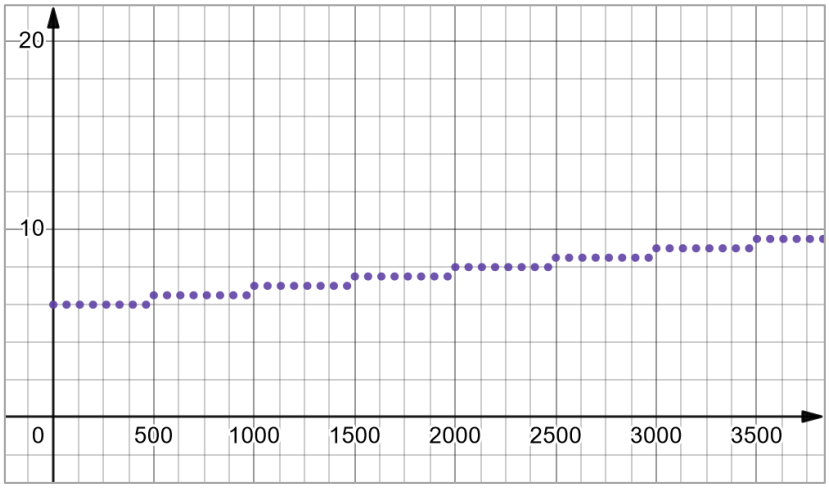


Figura 8. Grafico funzione costoDefaultTask.
https://www.desmos.com/calculator/2epakrbyrj

Decisioni di progetto

Le librerie utilizzate in questa sezione sono Selenium per la creazione di un'istanza del browser Firefox che contiene un'interprete JS. Quest'ultimo è utile per calcolare le coordinate di un nodo e inserirle in un dizionario. Ovviamente ciò è fattibile solo dopo aver renderizzato la pagina ed eseguito del codice JS. Questo viene fatto mediante funzione **`_create_driver(width, height)`** ove **`width`** e **`height`** sono inizializzate a 1600 e 900 (la mia risoluzione schermo).

La libreria Beautifulsoup è stata usata per il parsing del codice sorgente e la definizione di una funzione ricorsiva di creazione del NDOM. In merito a Beautifulsoup, è necessario scegliere il parser **`lxml`** o **`html5lib`**, visto che sono gli unici in grado di gestire eventuali tag non chiusi. Prima di utilizzare questa libreria comunque, è stato fatto un preprocessing del codice sorgente che rimuove i commenti, spazi inutili e i tag proibiti.

Si è pensato di programmare il modello NDOM come una classe Python avente la seguente interfaccia (a sinistra). Al centro c'è la lista di tag HTML che vengono rimossi prima ancora di iniziare la costruzione del NDOM, a destra ci sono i tag HTML che si possono assumere nodi foglia e nodi interni.

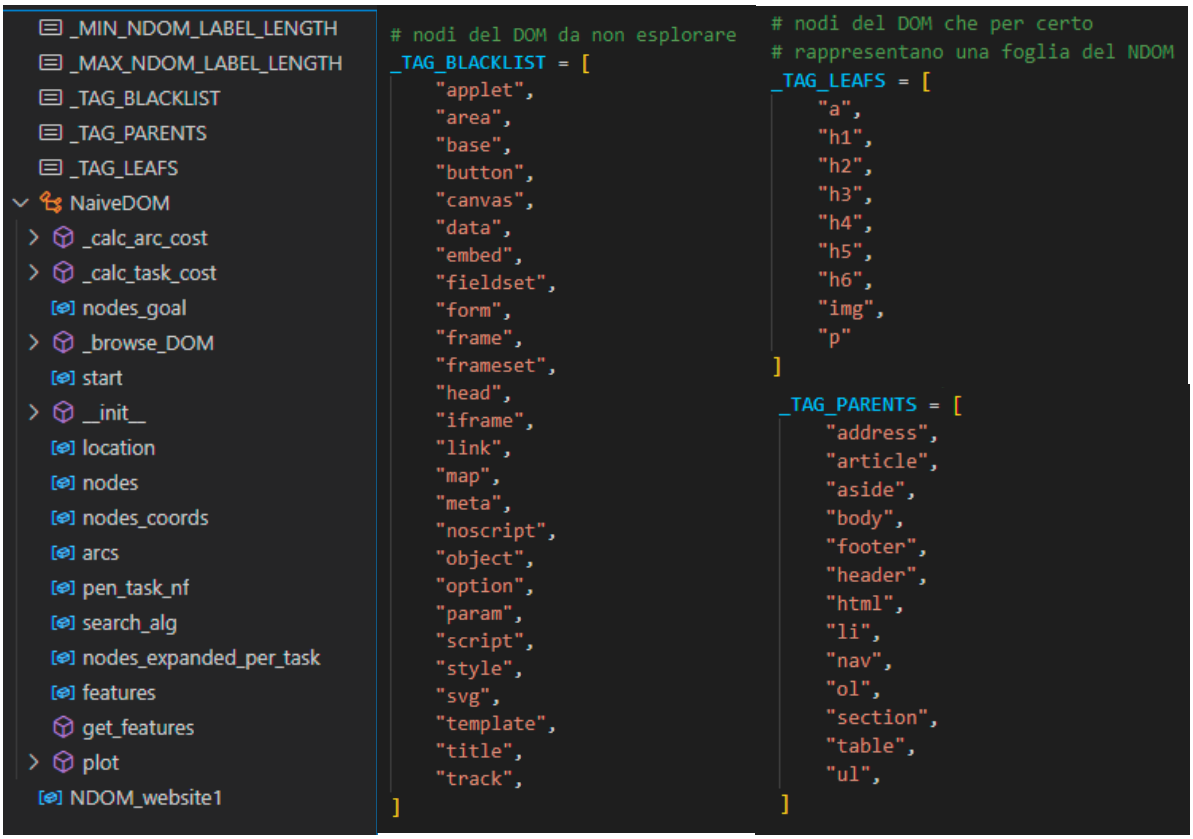


Figura 9.

Il metodo **`_browse_DOM`** è quello adibito alla costruzione del NDOM ed è presentato in pagina successiva. Si sfoglia il codice sorgente in modalità DFS, si stabilisce se un nodo è il nodo radice/interno/foglia del NDOM e si ottiene la sua label. Una volta esaminato il nodo corrente ed aver individuato i suoi nodi figli, si invoca una chiamata ricorsiva.

```
is_DOM_root = is_DOM_tag and not DOM_parent_xpath and not NDOM_parent_xpath
if is_DOM_root:
    # root e' la radice del NDOM
    xpath = f"//{root.name}"
    label = root.name

    next_NDOM_parent_xpath = xpath
    self.start = xpath

elif is_DOM_tag and root.name in _TAG_PARENTS:
    # root e' un nodo interno del NDOM
    xpath = f"{DOM_parent_xpath}/*[{(DOM_ci)}]"
    label = root.name

    next_NDOM_parent_xpath = xpath

elif is_DOM_tag and root.name in _TAG_LEAFS:
    # root e' un tag foglia
    xpath = f"{DOM_parent_xpath}/*[{(DOM_ci)}]"
    label = " ".join(root.stripped_strings)

    # elimino nodi che contengono testo lungo o non leggibile
    label_len = len(label)
    if label_len <= _MIN_NDOM_LABEL_LENGTH or label_len >= _MAX_NDOM_LABEL_LENGTH:
        return

elif is_DOM_tag:
    # root e' un tag che non e' ne' nodo interno del NDOM ne' foglia del NDOM
    xpath = f"{DOM_parent_xpath}/*[{(DOM_ci)}]"

    next_NDOM_parent_xpath = NDOM_parent_xpath

else:
    # isinstance(root, NavigableString)
    # root e' una stringa, quindi una foglia
    xpath = DOM_parent_xpath
    label = repr(root)

    # elimino nodi che contengono testo lungo o non leggibile
    label_len = len(label)
    if label_len <= _MIN_NDOM_LABEL_LENGTH or label_len >= _MAX_NDOM_LABEL_LENGTH:
        return
```

```
# sfoglio ciascun sottoalbero
if next_NDOM_parent_xpath:
    i = 1
    for child in root.children:
        self._browse_DOM(
            child,
            DOM_parent_xpath=xpath,
            DOM_ci=i,
            NDOM_parent_xpath=next_NDOM_parent_xpath,
            depth=next_depth,
            driver=driver,
        )
        # le stringhe dentro un tag non si considerano elementi figli del tag
        if not isinstance(child, (NavigableString, Comment)):
            i += 1
```

Come possiamo vedere dall'interfaccia in Figura 9 (sinistra), oltre agli attributi che descrivono la struttura del modello (**`nodes_goal`**, **`start`**, **`location`**, **`nodes`**, **`nodes_coords`**, **`arcs`**), c'è l'attributo **`features`** (dizionario delle 13 feature dette prima) e gli attributi **`search_alg`** e **`nodes_expanded_per_task`**.

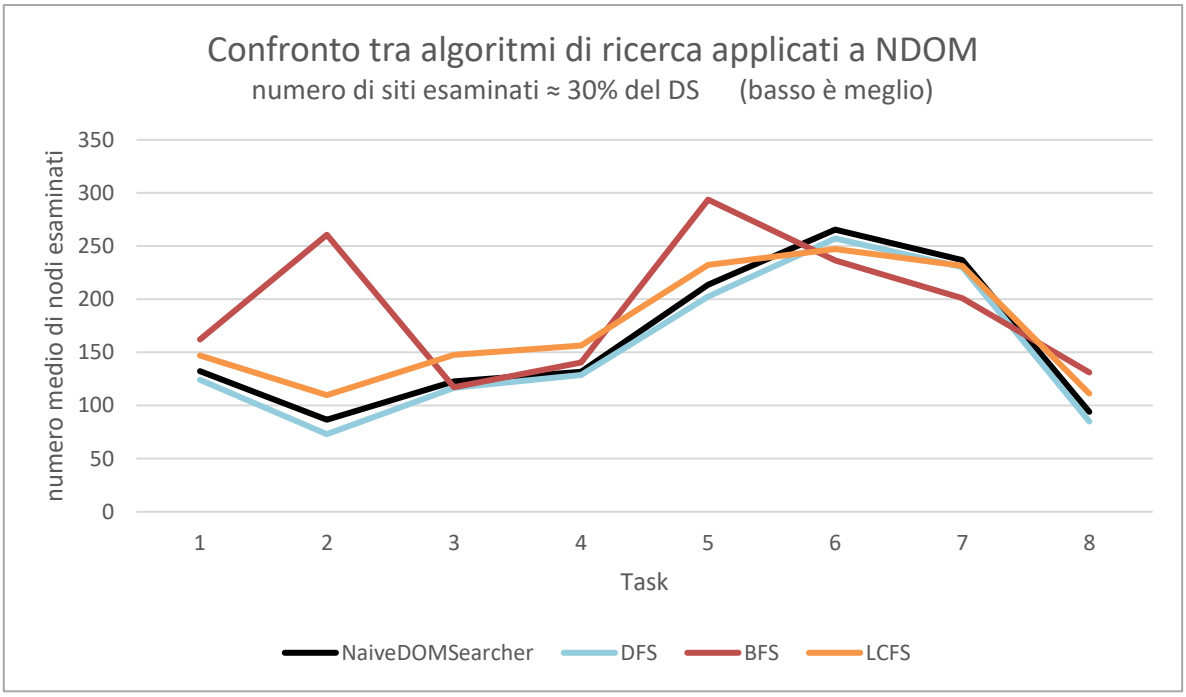
`search_alg` è una stringa che identifica uno tra gli algoritmi di ricerca di un Task non informati che è possibile applicare: "NaiveDOMSearcher", "DFS", "BFS", "LCFS". Per poter garantire questa funzionalità è stata modificata la classe **`Searcher`** della libreria AIPython aggiungendo al costruttore il parametro **`algorithm`** e modificando i metodi chiamati al momento dell'inserimento/rimozione di un percorso in frontiera. E' un attributo che risulta utile per il confronto degli algoritmi (sezione successiva).

`nodes_expanded_per_task` è un attributo (dizionario) auto-esplicativo: per un dato algoritmo di ricerca impiegato, associa ad ogni Task il numero di nodi che si sono esaminati prima di giungere a un nodo obiettivo.

Valutazione

Codice: **`/agent/ndom/benchmark.py`**

In questa sezione valutiamo l'algoritmo di ricerca costruito mettendolo a confronto con altri algoritmi non informati DFS, BFS e LCFS. Consideriamo un insieme di indirizzi web rappresentanti il 30% del DS. Successivamente costruiamo automaticamente 4 NDOM per ciascun sito, ciascuno dei quali usa un algoritmo di ricerca dei nodi obiettivo diverso. Il numero di nodi esaminati per ogni NDOM (e per ogni Task) è salvato nel file **`/agent/ndom/benchmark/benchmark_graph.xlsx`**.



Il grafico mostra come per il Task 1 (Circolari), tutti gli algoritmi esaminano in media lo stesso numero di nodi prima di giungere a un nodo obiettivo. Per il Task 2 (Organigramma) la situazione è diversa: potremmo ipotizzare che un nodo obiettivo possa trovarsi a una profondità maggiore, e quindi l'algoritmo BFS perde tempo esplorando l'albero in larghezza. Questo difetto della ricerca BFS non viene però assorbito dalla sua efficienza al Task 7 (circa 40 nodi in meno esaminati rispetto agli altri algoritmi), per cui BFS è da scartare.

Possiamo assumere che l'algoritmo che abbiamo costruito (linea nera) è da considerarsi una migliore alternativa al LCFS, anche a fronte del fatto che può sfruttare una complessità polinomiale di spazio e tempo. Seppur NaiveDOMSearcher impiega due frontiere (una PQ per i percorsi con profondità < 2 e uno stack LIFO), la prima di queste non desta problemi ed ha complessità di tempo trascurabile, in quanto dipende solamente dalla profondità del livello successivo alla radice. E' improbabile infatti che i template dei siti web dispongano tutti gli elementi come figli diretti del **<body>**.

Apprendimento Supervisionato

Sommario

La rappresentazione tramite modello NDOM discussa nella sezione precedente ci ha permesso, di fatto, di ingegnerizzare e aggiungere al DS iniziale 13 nuove feature. In questa sezione costruiamo e valutiamo dei modelli di apprendimento supervisionato (SL) che possano predire il valore della feature target **`metric`**. Si impiegheranno diversi approcci, come l'approccio classico, l'approccio Case-Based e l'approccio con metodi Ensemble.

Strumenti utilizzati e Decisioni di progetto

Vedere: **`/agent/models/nb_supervised_learning.ipynb`**

Queste due sezioni sono trattate separatamente nel file indicato perché richiedono l'esecuzione di codice.

Valutazione

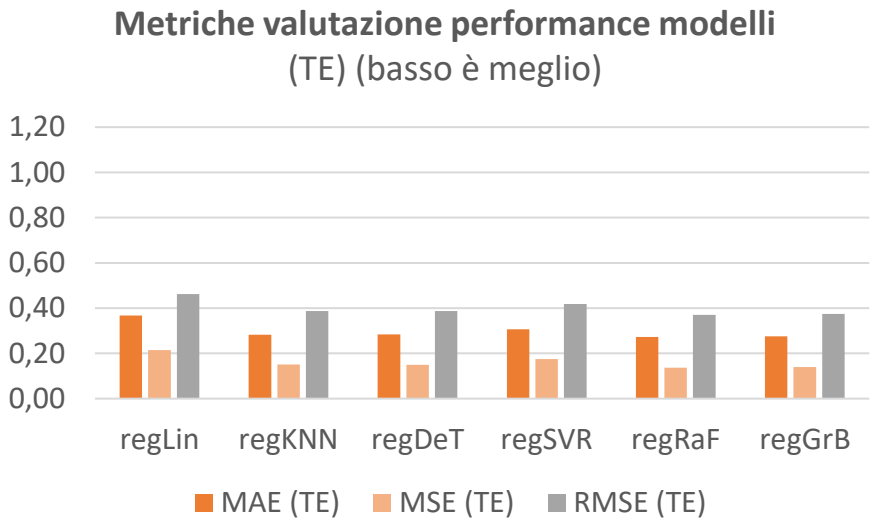
Vedere: **`/agent/models/charts/charts.xlsx`**

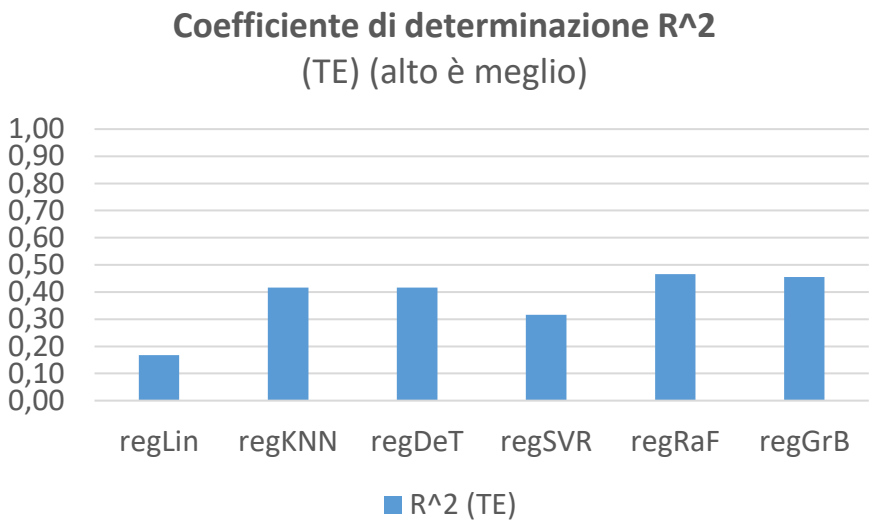
Per questo task di regressione sono state utilizzate diverse metriche, calcolate sia sui dati di training che sui dati di test.

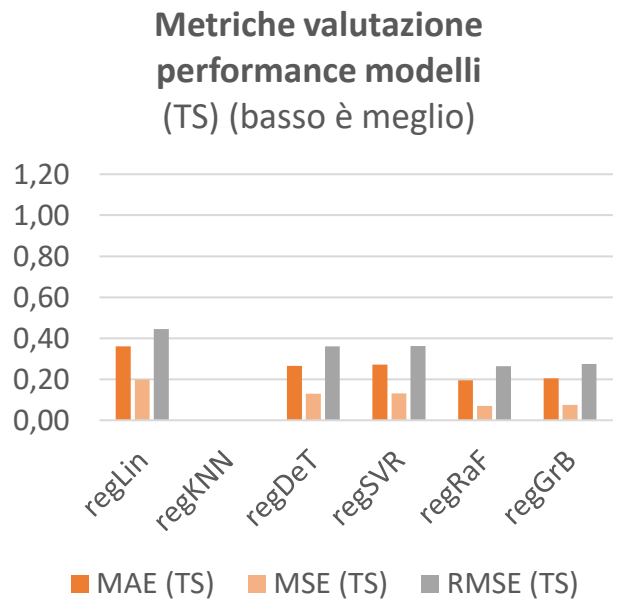
1. Mean Average Error (MAE): media delle differenze assolute tra le previsioni e i valori reali. A differenza del MSE, il MAE non penalizza tanto gli errori grandi, rendendolo una metrica più robusta alla presenza di valori anomali.
2. Mean Squared Error (MSE): misura la media dei quadrati delle differenze tra i valori previsti e i valori reali. Penalizza in modo più severo gli errori grandi rispetto a quelli piccoli.
3. Root Mean Squared Error (RMSE): Il RMSE è semplicemente la radice quadrata del MSE. Questa metrica è particolarmente utile quando si desidera interpretare l'errore nel contesto delle variabili originali, dato che riporta l'errore alla stessa unità di misura delle variabili stesse.
4. Coefficiente di Determinazione (R^2): Questa metrica fornisce una misura di quanto bene le previsioni del modello si adattano ai dati reali. Un R^2 di 1 indica che il modello è in grado di prevedere perfettamente i dati, mentre un $R^2 = 0$ indica che il modello non è in grado di prevedere i dati meglio di un modello costante.

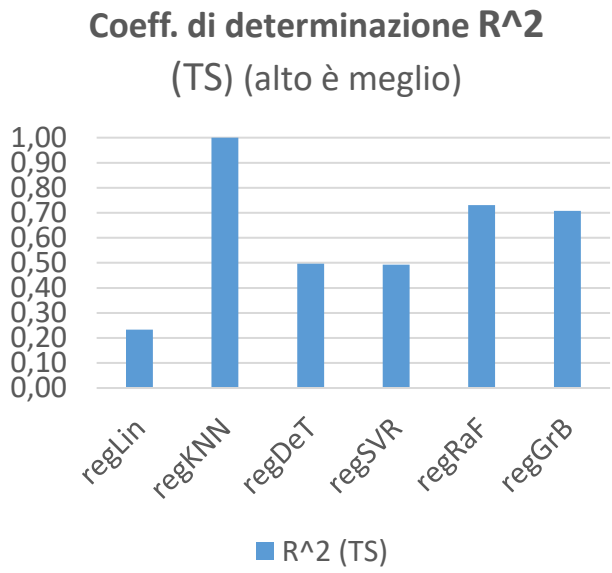
Di seguito verranno presentate e commentate le performance dei migliori modelli.

Modello	Combinazione ottimale di iperparametri	MAE (TS)	MSE (TS)	RMSE (TS)	R^2 (TS)	MAE (TE)	MSE (TE)	RMSE (TE)	R^2 (TE)
regLin		0,360450	0,198498	0,445513	0,232727	0,367821	0,214031	0,462326	0,167726
regKNN	{'algorithm': 'kd_tree', 'n_neighbors': 12, 'weights': 'distance'}	0,000000	0,000000	0,000000	1,000000	0,282082	0,150461	0,387708	0,416453
regDeT	{'criterion': 'friedman_mse', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 40, 'random_state': 1, 'splitter': 'random'}	0,266001	0,130166	0,360735	0,496544	0,283585	0,150130	0,386581	0,416471
regSVR	{'C': 100, 'epsilon': 0.2, 'gamma': 'scale', 'kernel': 'rbf'}	0,272003	0,131184	0,362131	0,492899	0,306309	0,175513	0,418576	0,316725
regRaF	{'bootstrap': True, 'criterion': 'friedman_mse', 'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 100, 'random_state': 66}	0,194428	0,069630	0,263857	0,730862	0,272786	0,137379	0,370500	0,466306
regGrB	{'criterion': 'friedman_mse', 'learning_rate': 0.05, 'max_depth': 10, 'min_samples_leaf': 20, 'min_samples_split': 120, 'n_estimators': 100, 'random_state': 66}	0,203744	0,075641	0,275016	0,707577	0,275188	0,139881	0,373839	0,455710


Metriche valutazione performance modelli
(TE) (basso è meglio)
1,20
1,00
0,80
0,60
0,40
0,20
0,00
regLin
regKNN
regDeT
regSVR
regRaF
regGrB
MAE (TE)
MSE (TE)
RMSE (TE)


Coefficiente di determinazione R^2
(TE) (alto è meglio)
1,00
0,90
0,80
0,70
0,60
0,50
0,40
0,30
0,20
0,10
0,00
regLin
regKNN
regDeT
regSVR
regRaF
regGrB
R^2 (TE)


Metriche valutazione performance modelli
(TS) (basso è meglio)
1,20
1,00
0,80
0,60
0,40
0,20
0,00
regLin
regKNN
regDeT
regSVR
regRaF
regGrB
MAE (TS)
MSE (TS)
RMSE (TS)


Coeff. di determinazione R^2
(TS) (alto è meglio)
1,00
0,90
0,80
0,70
0,60
0,50
0,40
0,30
0,20
0,10
0,00
regLin
regKNN
regDeT
regSVR
regRaF
regGrB
R^2 (TS)

regLin. Il modello ha dimostrato un'accuratezza discreta, con punteggi di usabilità che differiscono in media di 3,6 punti decimali sia sul TS (0,360450) che sul TE (0,367821). E' un modello sicuramente da scartare perché ha un valore di R^2 insufficiente, che implica bassa capacità di generalizzazione. Non a caso i valori di R^2, sia sul TS che sul TE, sono i minimi registrati.

regKNN. Per il regressore KNN possiamo fare una prima osservazione: siamo certi del fatto che i siti con lo stesso `page_template` tendono ad avere una valutazione simile, salvo quando un sito stravolge il suo assetto grafico continuando a utilizzare il template (raro).
Se utilizziamo la configurazione:

- `K` pari a12
- esempi simili ponderati sulla distanza vettoriale

possiamo pensare che questo algoritmo sia quello preferibile, e infatti ottiene un MAE di 0 sul TS e 0,282082 sul TE, indicando previsioni accurate. Il MSE è anch'esso molto basso, anche sul TE (0,150461), il che suggerisce una buona gestione delle deviazioni.

regDeT. Per questo modello, gli iperparametri migliori sono risultati:

- criterio di riduzione delle impurità `friedman_mse`
- altezza massima dell'albero pari a 10
- attuazione dello splitting solo se la partizione corrente ha almeno 40 esempi.

Con un R^2 di 0,496544 sul TS e 0,416471 sul TE, questo modello ha mostrato un buon equilibrio tra overfitting e underfitting. Compie in media un errore assoluto di poco più 2,5 punti decimali (0,266001 sul TS e 0,283585 sul TE) e il MSE (0,130166 sul TS e 0,150130 sul TE) indica che il modello è in grado di gestire le deviazioni. Questo ci suggerisce che un modello Ensemble basato su alberi di decisione porterà a prestazioni migliori.

regSVR. La migliore configurazione del SVR è:

- Soft-Margin SVM (visto che C>0)
- funzione kernel gaussiana.

Risulta essere migliore solo del regressore lineare, visto che compie un errore assoluto pari a 3 punti decimali.

regRaF e **regGrB**: Questi due modelli Ensemble basati su alberi di decisione risultano avere prestazioni simili. Sul TS compiono un errore assoluto esiguo di 2 punti decimali mentre sul TE di 2,7 punti decimali, denotando una notevole precisione. Entrambi i modelli hanno presentato le performance più elevate in termini di R^2 (0,730862 sul TS e 0,466306 sul TE), mostrando che sono in grado di spiegare una significativa parte della varianza nei dati.

Si è osservato che i coefficienti di determinazione R^2 dei modelli si aggirano intorno a 0,45. Una possibile spiegazione risiede nel fatto che i modelli ragionano su un insieme di feature nessuna delle quali, presa singolarmente, è fortemente esplicativa. Ricordiamoci infatti che la feature `page_ungrouped_multim` è esclusivamente un fattore di decisione sul quale dipende la valutazione del ground truth, ed è stata volutamente esclusa dalle feature considerate (vedi sezione *Caricamento DS e Feature Selection*). Testando il caso opposto comunque, sono state misurate valori di R^2 più alti (in media pari a 0,86), ma così facendo staremmo "barando".

Il valore di R^2 di uno dei nostri modelli indica la proporzione di dispersione che tale modello riesce a spiegare, rispetto a un modello di base che ipotizza sempre il valore medio. Per quanto nei nostri modelli si aggiri intorno a 0,45, il fatto che sia maggiore di 0 è comunque un segno indicante un comportamento più "intelligente" rispetto a un modello baseline "stupido" che predice calcolando il valore medio.

Ragionamento relazionale, Web Semantico

Sommario

In questa sezione si affronta l'argomento del ragionamento relazionale, preferito rispetto al ragionamento proposizionale in quanto una Homepage è a tutti gli effetti un tipo di individuo su cui è possibile fare operazioni di logica del primo ordine.
Nelle sezioni presenti fino ad ora, la feature **`school_id`** non era mai stata presa in considerazione; ora invece si noterà come ad essa sono correlate una serie di informazioni non numeriche ma comunque importanti. Con il Web Semantico siamo in grado di implementare queste correlazioni.

Strumenti utilizzati

Codice: **`/agent/kb/`**

Utilizziamo innanzitutto una KB scritta in Prolog i cui fatti sono asseriti a partire dalle informazioni del dataset **`ds3_gt`**, ovvero quello che abbiamo utilizzato nella sezione del SL. I fatti si trovano nei file **`kb_shared_facts.pl`**, e **`jobx_clauses.pl`**. Le regole, invece, nel file **`kb_shared_rules.pl`**.

Le operazioni che intendiamo fare, tuttavia, possono richiedere informazioni non presenti nel **`ds3_gt`**, ma sparse in altre KB. Ad esempio, il codice catastale del comune in cui si trova la scuola associata alla Homepage non è esplicitata in nessun fatto. La potremmo recuperare in due modi: (a) trasformando il dataset **`ds1`** in fatti appositi oppure (b) accedendo alla KB remota offerta dal MIUR. Si è optato per la seconda opzione, e al termine del ritrovamento, si sono creati dei fatti nella KB locale.

L'interrogazione della KB remota avviene tramite la funzione **`query_miur_sparql(query, endpoint)`**. L'interprete Python e SWI-Prolog interagiscono con pyswip. Tutte le altre informazioni assenti anche nella KB remota, sono state ottenute dall'endpoint SPARQL di Wikidata.

Decisioni di progetto

Di seguito viene descritta l'interpretazione semantica a cui fanno riferimento le varie operazioni (Job) sulla KB. Successivamente, verrà descritto ciascun Job.

Interpretazione semantica

Individui: L'interpretazione del dominio è tale per cui gli individui sono una Pagina, una Scuola (intesa come corso di studio, ad es. liceo classico, istituto tecnico, …), un Istituto scolastico che comprende 1 o più scuole, e altri individui inerenti alla posizione geografica.

Termini generali (possono essere impiegati da tutti i Job)

schoolassoc(Url, School_ID)

Simbolo di funzione largamente utilizzato, crea un'associazione semantica pagina-scuola.

institute(Institute_ID, Institute_Name)

Simbolo di funzione che raggruppa l'ID e il nome di un istituto scolastico.

Predicati generali

```
page(schoolassoc(Url, School_ID),
    details(Width, Height, Load_time_ms, Template, Menu_or, Ungrouped_multim),
    ndom(NDOM_Nodes, NDOM_Height, NDOM_Tasks),
    Metric).
```

Vero quando tutti gli argomenti sono inerenti alla stessa pagina del dataset **`ds3_gt`**.

```
school_geofact(School_ID, city(...), province(...), region(...)).
```

Vero quando tutti gli argomenti sono informazioni geografiche corrette della scuola (primo argomento).

Job 1

Vedere: **`/agent/kb/jobs/job1_output.pl`**

Obiettivo: Per ciascuna Homepage che impiega un metodo non standard di redirect, creare un report indicante l'istituto scolastico a cui fa capo la scuola e raccolta dei contatti di tutte le scuole gestite da tale istituto. Questo Job può essere utile per avvertire simultaneamente i presidi degli istituti che condividono la stessa Homepage. Per farlo si sono seguiti i seguenti step.

1. Regola di rilevazione pagina: tutte le pagine che eseguono un redirect standard, cioè che rispondono al client HTTP con una risposta 301, vengono già rilevate e gestite nella fase di preprocessing. Le pagine che non seguono questa procedura hanno un codice sorgente privo di contenuti leggibili. Vengono rilevate osservando l'altezza e il numero di nodi del NDOM.

```
page_wrongly_redirects(schoolassoc(Url, School_ID)) :-
    page(schoolassoc(Url, School_ID), _, ndom(NDOM_Nodes, NDOM_Height, _), _),
    NDOM_Height =< 1,
    NDOM_Nodes =< 2.
```

2. Ricerca istituto scolastico + tutte le scuole gestite da quell'istituto: A partire dallo **`School_ID`** individuato, eseguiamo questa query.

```
PREFIX miur: <http://www.miur.it/ns/miur#>
Select ?CodiceIstitutoRiferimento ?DenominazioneIstitutoRiferimento ?CodiceScuola {
    graph ?g {
        ?S miur:CODICEISTITUTORIFERIMENTO ?CodiceIstitutoRiferimento.
        ?S miur:DENOMINAZIONEISTITUTORIFERIMENTO ?DenominazioneIstitutoRiferimento.
        ?S miur:CODICESCUOLA 'PNTL012017'.
        ?C miur:CODICEISTITUTORIFERIMENTO ?CodiceIstitutoRiferimento.
        ?C miur:CODICESCUOLA ?CodiceScuola.
    }
}
LIMIT 50
```

Ciascuna riga di questa query viene convertita in un fatto:

```
institute_has_school(institute(Institute_ID, Institute_Name), School_ID)
```

Come si può notare, nella KB vengono aggiunti solamente i fatti institute_has_school necessari, cioè solamente per le pagine di cui sappiamo per certo che violano il vincolo del redirect. Avremmo anche potuto chiamare la seguente query, ma ciò avrebbe creato un file di dimensione esagerata.

```
PREFIX miur: <http://www.miur.it/ns/miur#>
Select ?CodiceIstitutoRiferimento ?DenominazioneIstitutoRiferimento ?CodiceScuola {
    graph ?g {
        ?S miur:CODICEISTITUTORIFERIMENTO ?CodiceIstitutoRiferimento.
        ?S miur:DENOMINAZIONEISTITUTORIFERIMENTO ?DenominazioneIstitutoRiferimento.
        ?S miur:CODICESCUOLA ?CodiceScuola.
    }
}
LIMIT 50
```

3. Report parziale. Asseriamo quali sono gli istituti (ID, Nome e ID delle scuole associate) delle pagine che soddisfano il vincolo.

```
is_partial_report1(institute_with_all_schools(institute(Institute_ID, Institute_Name),
Institute_Schools_IDs), schoolassoc(Url, School_ID)) :-
    page_wrongly_redirects(schoolassoc(Url, School_ID)),
    institute_has_school(institute(Institute_ID, Institute_Name), School_ID),
    findall(S, institute_has_school(institute(Institute_ID, _), S), Institute_Schools_IDs).
```

4. Report finale. Asseriamo quali sono le informazioni di contatto di ogni scuola dell'istituto del passo 3. La query è simile a quella del punto 2:

```
PREFIX miur: <http://www.miur.it/ns/miur#>
Select ?CodiceScuola ?DenominazioneScuola ?IndirizzoScuola ?DescrizioneComune
?CapScuola ?IndirizzoEmailScuola {
    graph ?g {
        ?S miur:CODICESCUOLA 'PNTL012017'.
        ?S miur:CODICESCUOLA ?CodiceScuola.
        ?S miur:DENOMINAZIONESCUOLA ?DenominazioneScuola.
        ?S miur:INDIRIZZOSCUOLA ?IndirizzoScuola.
        ?S miur:DESCRIZIONECOMUNE ?DescrizioneComune.
        ?S miur:CAPSCUOLA ?CapScuola.
        ?S miur:INDIRIZZOEMAILSCUOLA ?IndirizzoEmailScuola.
    }
}
Limit 1
```

Il report finale è un fatto i cui argomenti sono tutti simboli di funzione. Il 3° argomento è una lista di simboli di funzione schoolcontact.

```
is_full_report_for_job1(
    schoolassoc(...),
    institute(...),
    [ schoolcontact(CodiceScuola, DenominazioneScuola, IndirizzoScuola, DescrizioneComune,
CapScuola, IndirizzoEmailScuola)  ...
    ]).
```

Job 2

Vedere: `/agent/kb/jobs/job2_output.pl`

Obiettivo: Per ciascuna Homepage che necessita un urgente miglioramento grafico, creare un report indicante l'istituto scolastico a cui fa capo la scuola e raccolta dei contatti di tutte le scuole gestite da tale istituto. Gli step seguiti sono uguali a quelli del Job 1, cambia solo la regola di rilevazione della pagina (Step 1), che ora si chiama page_need_improvement.

```
page_needs_improvement(schoolassoc(Url, School_ID), Treshold1, Treshold2) :-
    page(schoolassoc(Url, School_ID), details(_, _, _, Template, _, Ungrouped_multim), _, Metric),
    Ungrouped_multim >= Treshold1,
    Metric =< Treshold2,
    is_good_template(Good_Templates),
    \+ member(Template, Good_Templates).
```

Job 3

Vedere: `/agent/kb/jobs/job3_output.txt`

Obiettivo: stilare una classifica delle regioni italiane con più alta frequenza relativa di pagine con una buona metrica.

1. Regola di rilevazione pagina.

```
page_has_good_metric(schoolassoc(Url, School_ID)) :-
    page(schoolassoc(Url, School_ID), details(_, _, _, _, _, Ungrouped_multim), _, Metric),
    Metric > 3.8,
    Ungrouped_multim =< 6.
```

2. Asserzione della regione in cui è localizzata ciascuna scuola.

```
school_is_in_place(School_ID, Place) :-
    school_geofact(School_ID, _, _, region(Place)).
```

3. Asserzione della frequenza relativa di buone pagine tra tutte quelle della regione. Un'area geografica (Place) ha una certa frequenza relativa quando questo valore è il rapporto tra numero di pagine con buona metrica (presenti sempre nell'area Place) e numero totale di pagine dell'area.

```
is_relative_frequency_for_place(Place, Relative_Frequency) :-
    findall(_,
            (school_is_in_place(School_ID, Place), page_has_good_metric(schoolassoc(_, School_ID))),
            List_Good_In_Place),
    is_list_length(List_Good_In_Place, Numerator),
    findall(_, (school_is_in_place(_, Place)), List_All_In_Place),
    is_list_length(List_All_In_Place, Denominator),
    Relative_Frequency is Numerator / Denominator..
```

Vedendo questa clausola, potremmo pensare che sia superfluo aggiungere il predicato school_is_in_place, visto che esiste già il fatto school_geofact e potremmo chiamare una qualsiasi variabile presente in esso con il nome Place. Così facendo, però, se volessimo applicare filtri più specifici all'area geografica, ciò implicherebbe il dover riscrivere clausole is_relative_frequency_for_place simili. L'importante è quindi garantire che la regola school_is_in_place, definita al punto 2, sia consultabile solamente per questo Job.

4. Asserzione della classifica.

Utilizziamo il simbolo di funzione place_rf per indicare una tupla (Area geografica, Frequenza relativa di siti buoni). La relazione d'ordine su di esso è equivalente a quella sui numeri reali, visto che si considera unicamente la frequenza relativa.

```
place_order(<, place_rf(_, Relative_Frequence1), place_rf(_, Relative_Frequence2)) :-
    Relative_Frequence1 =< Relative_Frequence2.

place_order(>, place_rf(_, Relative_Frequence1), place_rf(_, Relative_Frequence2)) :-
    Relative_Frequence1 > Relative_Frequence2.
```

La classifica che vorremmo calcolare è una lista Rank tale per cui ogni elemento è un simbolo di funzione (tupla) place_rf. Il primo termine è Place, che deve appartenere alla lista (senza duplicati) di tutti i Place in cui sono mappate tutte le pagine. Il secondo è la frequenza relativa. La classifica è valida se tutti gli elementi sono ordinati in modo decrescente.

```
is_rank_of_places(Rank) :-
    findall(X, school_is_in_place(_, X), List_Places_W_Dups),
    setof(Y, member(Y, List_Places_W_Dups), List_Places_WO_Dups),
    bagof(place_rf(Place, Relative_Frequency),
            (member(Place, List_Places_WO_Dups),
              is_relative_frequency_for_place(Place, Relative_Frequency)
            ), Unordered_Rank),
    predsort(place_order, Unordered_Rank, Rank_Ascendant),
    reverse(Rank_Ascendant, Rank).
```

Job 4

Vedere: `/agent/kb/jobs/job4_output.txt`

Obiettivo: stilare una classifica delle province italiane (solamente quelle che hanno come "capitale" il capoluogo della regione) con più alta frequenza relativa di pagine con una buona metrica.

1. Regola di rilevazione della pagina: uguale a quella del Job 3.

2. Asserzione della provincia in cui è localizzata ciascuna scuola.

```
school_is_in_place(School_ID, Place) :-
    school_geofact(School_ID, province(Place)).
```

Anche in questo caso, dobbiamo precisare che il fatto school_geofact viene creato solo dopo aver controllato il criterio della provincia che ci siamo posti, per evitare di aggiungere fatti che non verranno sfruttati.
school_geofact comunque viene asserito grazie all'interoperabilità semantica tra due KB. In questo caso si compone di ID scuola e nome della provincia. Il nome della provincia è ottenibile a partire dal codice catastale della scuola seppur queste due informazioni risiedono in due KB diverse. Questo è possibile in quanto la proprietà miur:CODICECOMUNESCUOLA ha stesso significato di wdt:P806.

Questa fase viene svolta interrogando l'endpoint di Wikidata e sottoponendo la query seguente.

```
SELECT ?ProvinciaLabel
WHERE {
    ?Citta wdt:P806 "G888";
    wdt:P31 wd:Q747074;
    wdt:P131 ?Provincia.
    ?Provincia wdt:P131 ?Regione.
    ?Regione wdt:P36 ?CapoluogoDiRegione.
    ?Provincia wdt:P36 ?CapoluogoDiRegione.

    SERVICE wikibase:label { bd:serviceParam wikibase:language "it,en". }
}
LIMIT 1
```

Se questa restituisce un risultato, la scuola rientra nel vincolo, altrimenti no. In sintesi, l'interazione tra le 3 KB (quella locale, quella remota del MIUR e quella di Wikidata) avviene come indicato in Figura 10.

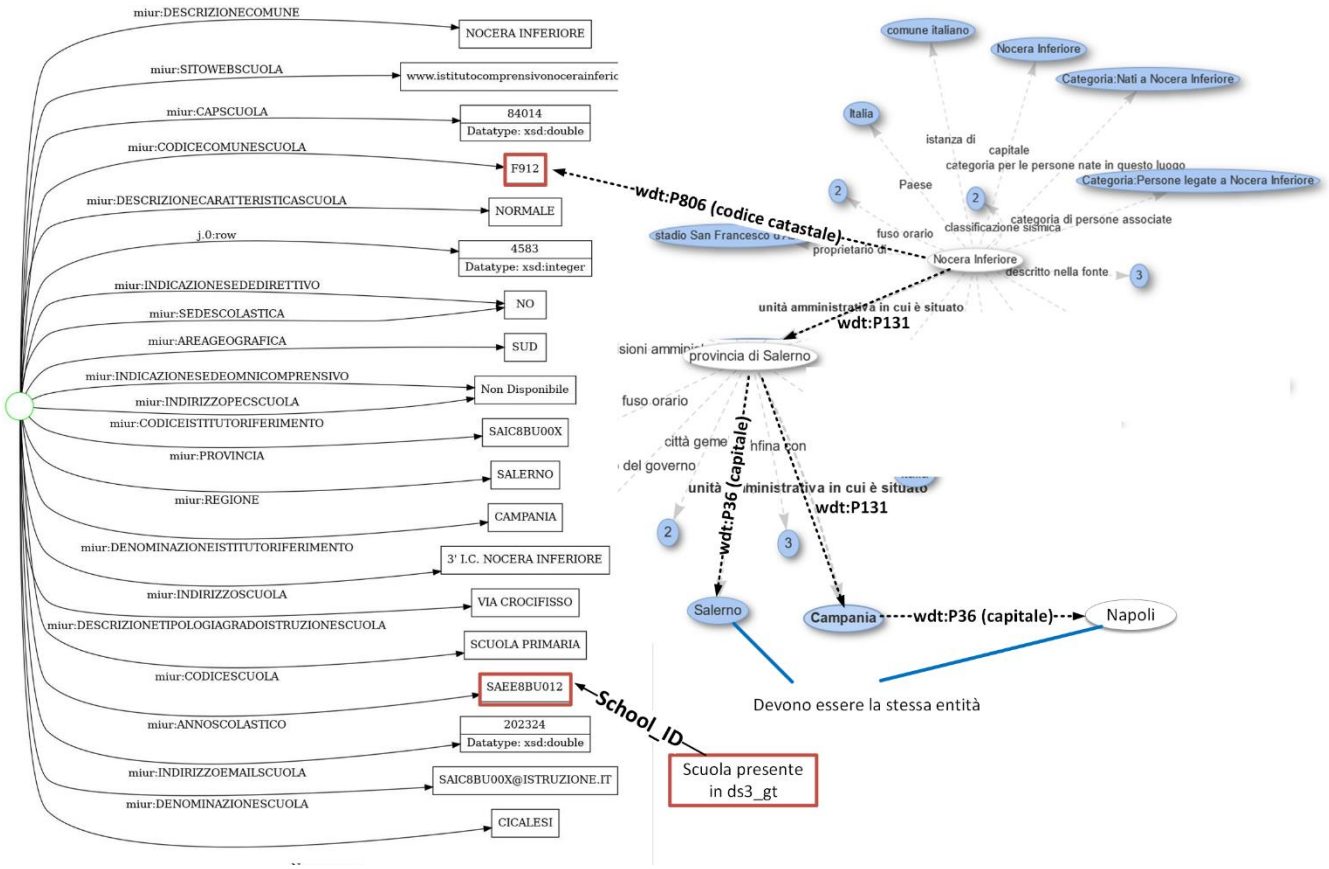


Figura 10. Navigazione nel web semantico per il Job 4.

3. Asserzione della frequenza relativa di buone pagine tra tutte quelle della regione.

4. Asserzione della classifica. Questi due step sono uguali a quelli del Job 3.

Modello probabilistico: Rete Bayesiana

Sommario

Nella fase iniziale di questo progetto, legata alla creazione del Grund Truth e alla scelta di feature da inserire nei dataset, è stato possibile visionare tutti i template disponibili e tutte le Homepage, e pertanto si è potuta intuire l'esistenza di dipendenze tra le varie features. I modelli grafici probabilistici, mediante i task di inferenza probabilistica che possiamo fare su di essi, ci danno informazioni utili.

Strumenti utilizzati

Il modello grafico usato è una Rete Bayesiana, cioè un DAG che rappresenta un insieme di variabili aleatorie con le loro dipendenze condizionali. I nodi sono le variabili, gli archi rappresentano la dipendenza condizionata. La probabilità condizionata dei valori assumibili da un nodo (a partire da quelli dei suoi genitori) è espressa in una CPT. Tutte le operazioni sulla BN sono state fatte con la libreria Python pgmpy.

Decisioni di progetto

Vedere: `/agent/pgm/bn_creator.py`

Innanzitutto, poiché la libreria pgmpy non supporta l'apprendimento di parametri e l'inferenza su variabili aventi dominio continua, si è adoperata una discretizzazione del dataset, il cui mapping è descritto nel dizionario `DS_DISCRETE_MAPPING_DEFAULT`. Nelle sezioni seguenti sono elencati in ordine i passi seguiti per la realizzazione della BN e l'inferenza.

Struttura della BN

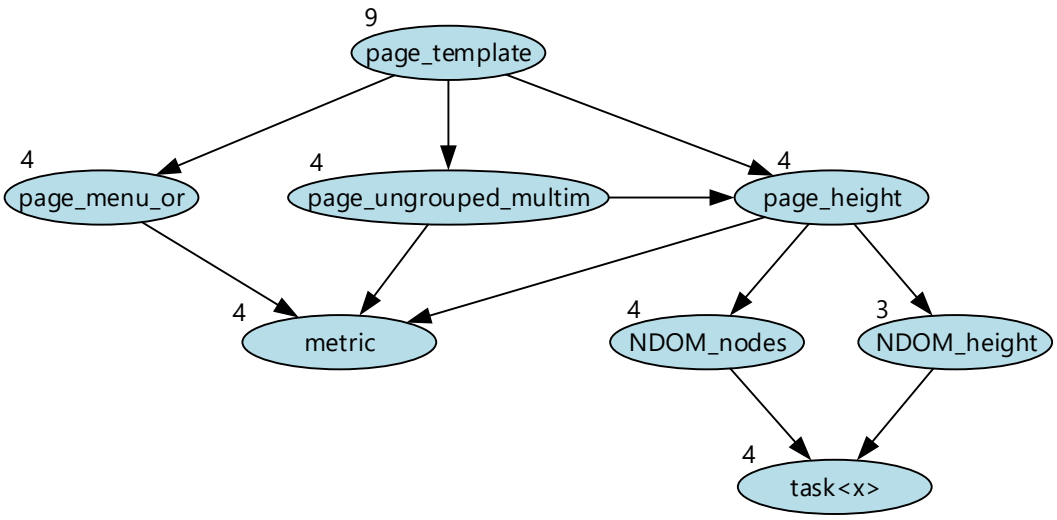


Figura 11. BN + cardinalità variabili.
La struttura non è stata appresa dal DS, ma progettata, come riportato nel sommario.

In generale, quando osserviamo un template (ad es. uno tra quelli in Figura 2), possiamo notare come questo decide l'orientamento del menu della pagina e influenza in parte anche l'altezza della pagina stessa. Inoltre, sarebbe corretto dire che ciascun template contribuisce in maniera differente anche al numero di

elementi non raggruppati? Sì perché ad esempio le pagine con template #8 sono tutte caratterizzate da una sezione non intitolata avente numero vario di elementi confusionari (Figura 10 e 11).

Da quali variabili dovrebbe dipendere la feature `metric`? Di certo sappiamo che l'utente non è a conoscenza del Template ID, quindi non assegna un punteggio in funzione di esso; piuttosto osserva ciò che *dipende* dal template: l'orientamento del menu, altezza della pagina e numero di elementi confusionari.

Venendo ora alle variabili **`NDOM_nodes`** e **`NDOM_height`**, entrambe sono indipendenti (può esistere un NDOM con tanti nodi ed altezza 1 o viceversa). Non vengono influenzate dal template della pagina ma solo da un'astrazione di essa, cioè solamente dall'altezza.

Le variabili **`page_load_time_ms`** e **`page_width`** non sono invece incluse nella BN perché si sono assunte indipendenti da tutte le altre.

Figura 12. https://www.patettacairo.edu.it/

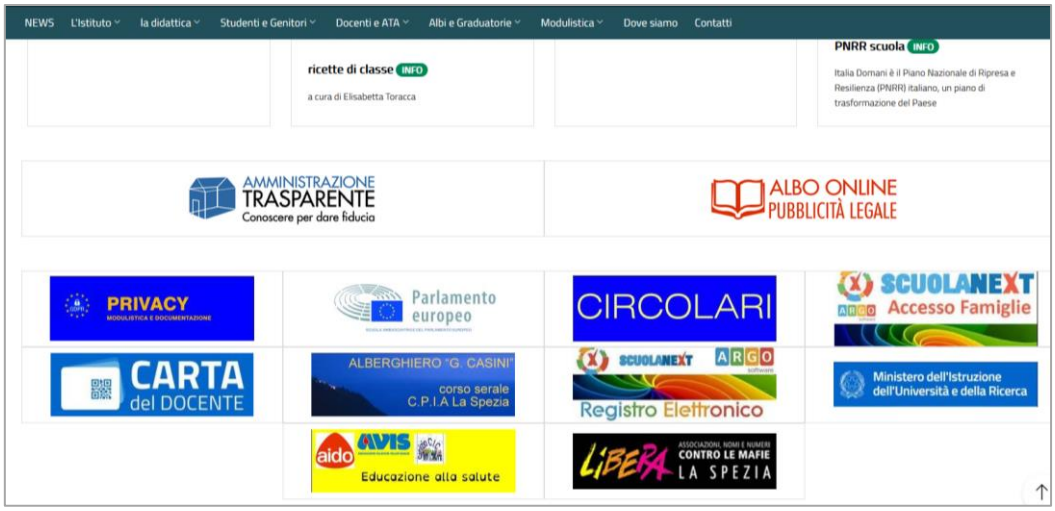


Figura 13. https://www.alberghierolaspezia.edu.it/

Apprendimento parametri della BN

Vedere: `/agent/pgm/bif/bn_estimator_MLE.bif`
`/agent/pgm/bif/bn_estimator_BDeu.bif`

Nel caso della nostra BN, un parametro è semplicemente una cella di una CPT, cioè la probabilità che una variabile aleatoria (=un nodo) assuma un determinato valore a partire da una combinazione di valori delle variabili genitore. Eventualmente, il numero di parametri può essere ridotto, ad es. quando un parametro può essere ricavato dagli altri, associati sempre alla stessa combinazione di variabili genitore. Non ci preoccuperemo di questa azione, in quanto ciò viene fatto automaticamente dalla libreria pgmpy.

Un primo approccio all'apprendimento dei parametri è l'utilizzo di uno **stimatore di massima verosimiglianza (MLE),** che genera una lista di parametri $\boldsymbol{\theta}$ tali da massimizzare la funzione di verosimiglianza $L(\boldsymbol{\theta}|DS)$, cioè la probabilità che il DS sia stato generato proprio con questa configurazione di parametri. In simboli:

$$\boldsymbol{\theta_{MLE}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, L(\boldsymbol{\theta}|DS)$$

In questo stimatore, un singolo parametro viene stimato contando (e normalizzando) il numero di occorrenze nel DS in cui il valore x della variabile aletaoria X compare associato a una particolare combinazione di variabili genitore (chiamate $parent(X)$):

$$\theta_X = \frac{\#(x,\ parent(X))}{\#parent(X)}$$

Lo stimatore MLE potrebbe ritenersi opportuno per il nostro progetto visto che il DS è fortemente esplicativo del ground truth. Tuttavia bisogna risolvere il problema delle probabilità nulle che riscontriamo quando il numeratore è 0, cioè quando esiste teoricamente una combinazione di valori di variabili che però non è mai registrata nel DS. Ad esempio, nel file `bn_estimator_MLE.bif`, in corrispondenza della CPT di `page_menu_or` è presente la seguente dicitura:

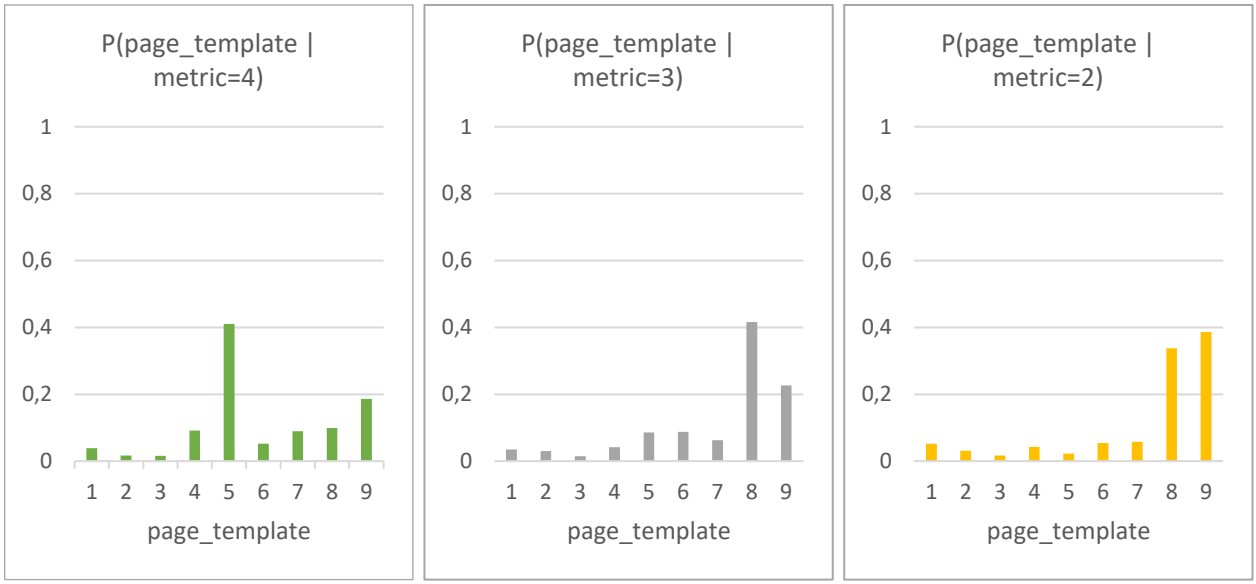
```
probability ( page_menu_or | page_template ) {
    …
    ( 9 ) 0.0, 0.5536, 0.0657, 0.3806;
    …
}
```

ad indicare che se un sito web non segue un template, allora è impossibile che non abbia nessun menu. Questo è abbastanza plausibile ma non lo sappiamo con certezza. Per risolvere questo problema, dobbiamo passare a uno stimatore Bayesiano che esprimono il nostro belief sulle variabili prima ancora di osservare il DS. La libreria offre [5] uno stimatore Bayesiano con configurazione `prior_type="BDeu"` + `equivalent_sample_size = 20` tale per cui ogni parametro è visto come variabile aleatoria che segue la distribuzione di Dirichlet. Gli pseudo-conteggi equivalgono ad aver osservato 20 campioni uniformi di ciascuna variabile (e ciascuna configurazione genitore).

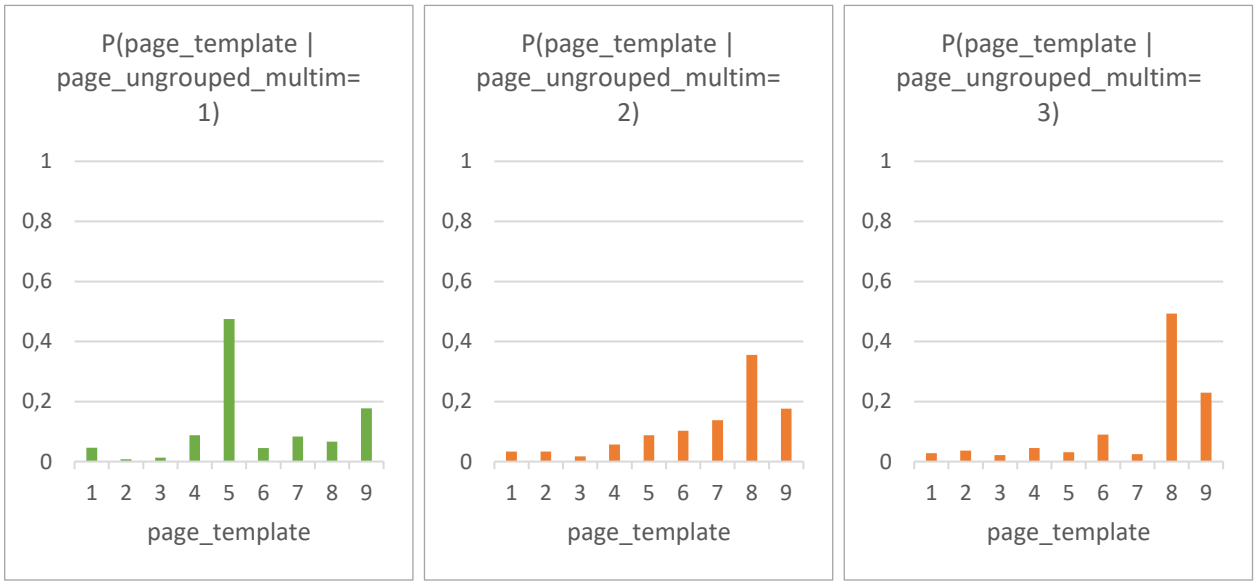
Query

Riguardo i task di inferenza probabilistica (per rispondere alle query sottoposte) si è deciso di utilizzare il metodo di inferenza esatta basato su eliminazione di variabili, a fronte del limitato numero di righe del dataset che garantisce un normale tempo di esecuzione. Le query da sottoporre alla BN sono elencate nella lista `BN_QUERIES_DEFAULT`. Si dividono in 4 tipi:

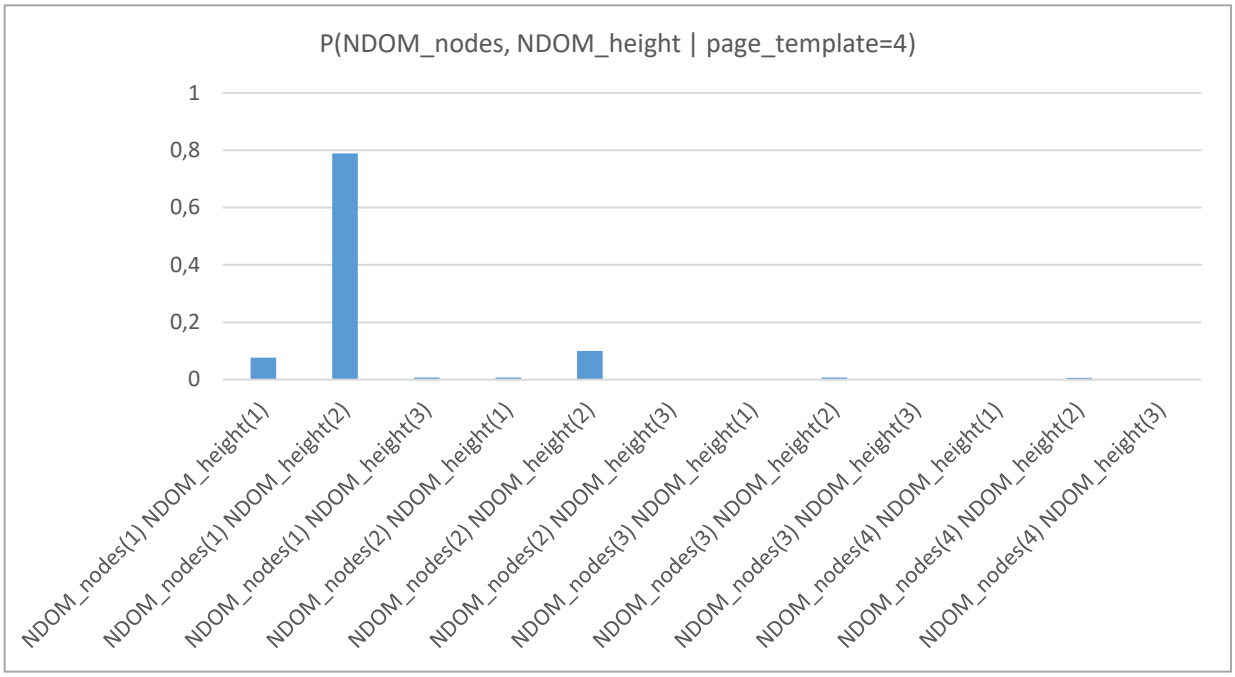
1. Dato il punteggio di usabilità, conoscere la probabilità che tale pagina segua un determinato template. $P(page\_template \mid metric = x)$



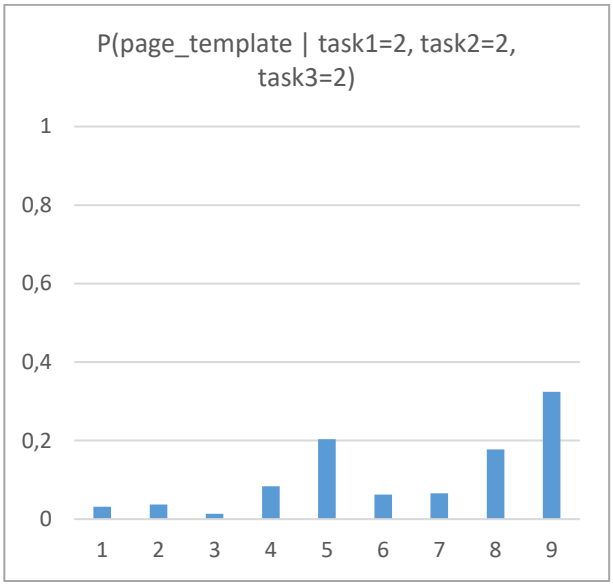
2. Dato il numero di elementi non raggruppati della pagina, conoscere la probabilità che tale pagina segua un determinato template.
$P(page\_template \mid page\_ungrouped\_multim = x)$

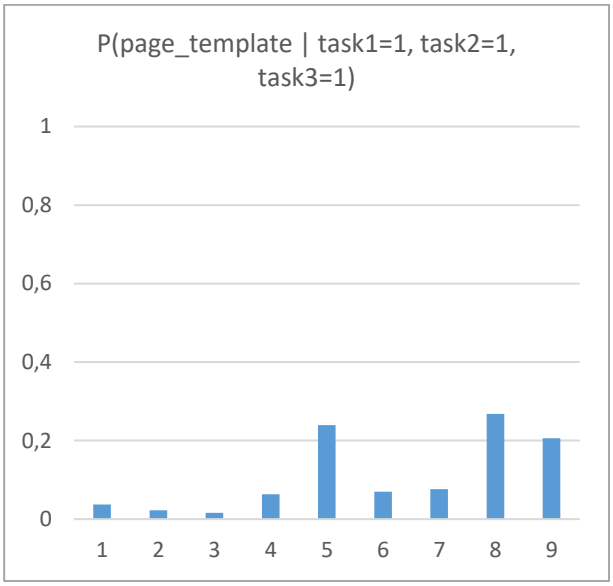


3. Dato il template, conoscere con che probabilità la pagina ha un numero di nodi del NDOM e rientri in una fascia di altezza. $P(NDOM\_nodes,\ NDOM\_height \mid page\_template = x)$



4. Data la fascia dei punteggi Task1, Task2 e Task3, conoscere con che probabilità la pagina segua un determinato template. $P(page\_template\ |task1 = x, task2 = y, task3 = z)$





ds

Conclusioni

Per questioni di tempo, durante la realizzazione di questo progetto non sono stati utilizzati strumenti che potessero ricavare automaticamente il valore della feature **`page_ungrouped_multim`**, il che può essere fatto per esempio, usando tecniche di Webpage Segmentation (applicate sullo screenshot della pagina) che coinvolgono algoritmi di Clustering come DB-SCAN [6] , k-means o VIPS [7].

L'ambito di questo progetto comunque, può essere esteso analizzando tutti i siti del dataset **`ds1`** e catalogando i risultati per tipologia di scuola.

Bibliografia

[1] [Online]. Available: https://en.wikipedia.org/wiki/Heuristic_evaluation#Nielsen's_heuristics.

[2] [Online]. Available: https://www.w3.org/TR/WCAG21/.

[3] [Online]. Available: https://en.wikipedia.org/wiki/System_usability_scale.

[4] [Online]. Available: https://en.wikipedia.org/wiki/XPath.

[5] [Online]. Available: https://pgmpy.org/param_estimator/bayesian_est.html.

[6] [Online]. Available: https://github.com/lqtri/WebPage-Segmentation--WPS-.

[7] [Online]. Available: https://github.com/tpopela/vips_java.

[8] [Online]. Available: https://artint.info/3e/html/ArtInt3e.Ch9.S5.html.