UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO
DIPARTIMENTO DI INFORMATICA

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-745751

Sommario

Introduzione

Idea del progetto

L'idea di fondo è l'applicazione di alcuni metodi di Ingegneria della Conoscenza su un dominio di interesse, l'usabilità di una pagina web. Si va prima a circoscrivere un ambito di riferimento, che nel nostro caso, è l'insieme delle Homepage delle scuole superiori pubbliche italiane (aggiornate a novembre 2023).

Metriche di usabilità già esistenti

Le metriche rilevanti che potrebbero essere applicate in questo contesto sono le Euristiche di Nielsen [1] e le WCAG 2.1 [2] per le quali però gli strumenti software ad essi correlati (qui elencati) non sono adatti 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 in 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**. Si approfondirà 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 (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) Riproduzione 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) Raccolta 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.

   3.3. Ricostruzione di alcune righe del dataset di partenza mediante creazione e ragionamento su un'**ontologia di dominio**.

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 (Figura 1). Per generalizzare (includendo qualsiasi contenuto multimediale, e quindi anche video) introduciamo la feature `page_ungrouped_multim`.

Figura 1.

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:

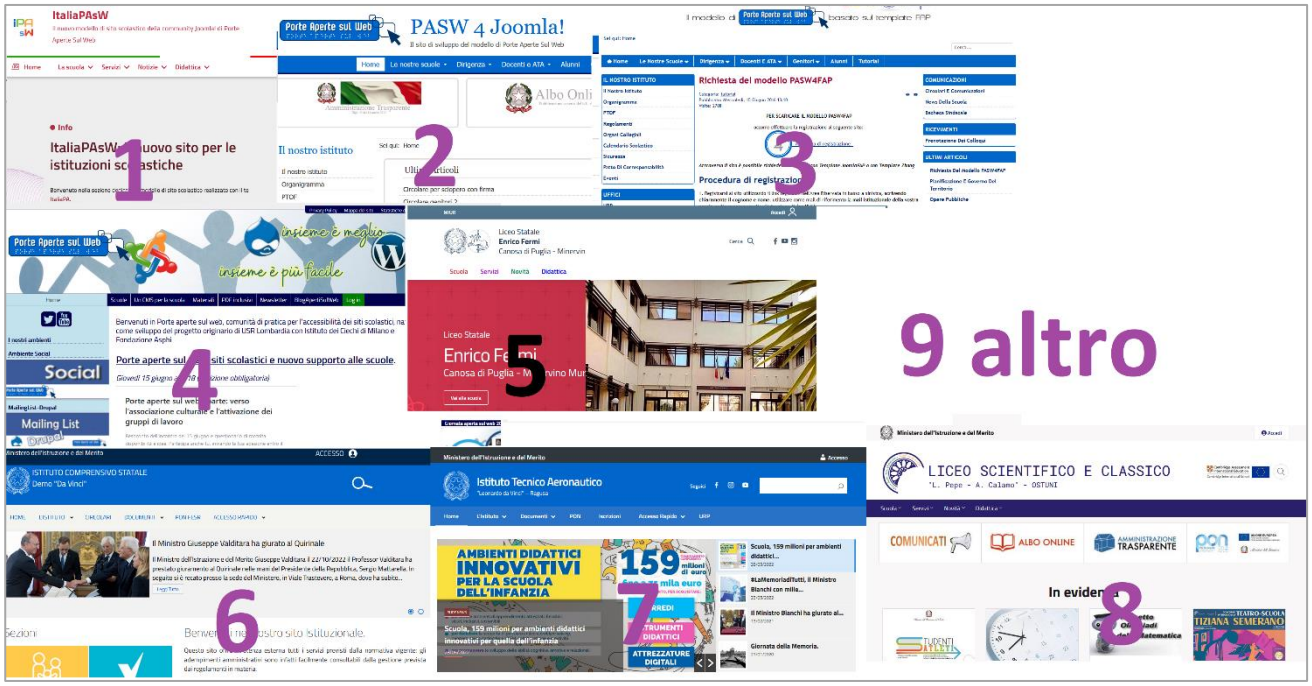


Figura 2.

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://web.spaggiari.eu/www/app/default/index.php?p=pvb&s=pvb (Esempio)

Preprocessing del dataset delle scuole

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

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`**: Link. 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 inserendo 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. Features = 6 feature (pag. 6) + 13 feature (pag. 10).
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:

- 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.

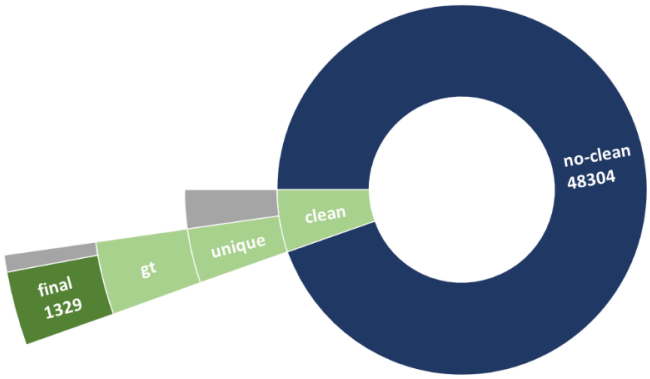


Figura 3. Proporzione del numero di elementi dei vari DS.

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 con 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.

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.

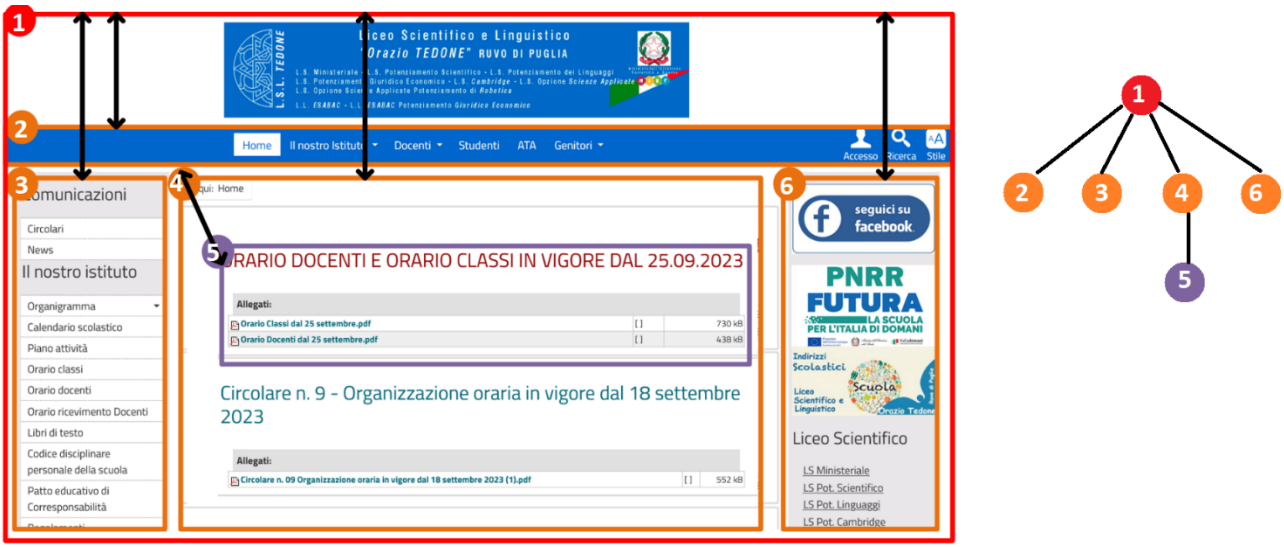


Figura 4. https://www.liceotedone.edu.it/ . Rappresentazione semplificata del NDOM.

Osservando lo screenshot di questo sito web (Figura 4), 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 anche perché visualizziamo una qualsiasi pagina web 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.

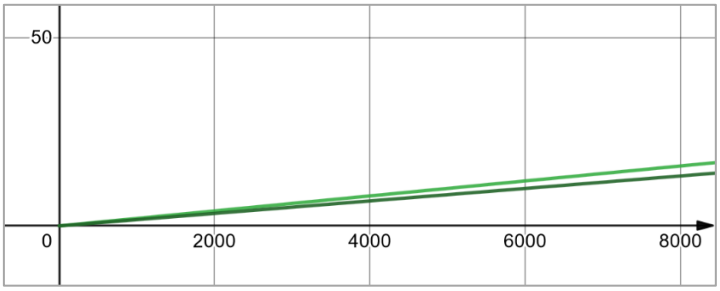


Figura 5. Grafico funzione $c(distanza)$ di costo dell'arco padre-figlio.
https://www.desmos.com/calculator/aaqy3tao8g

Calcolo di un task con algoritmo di ricerca

Codice: **`/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.

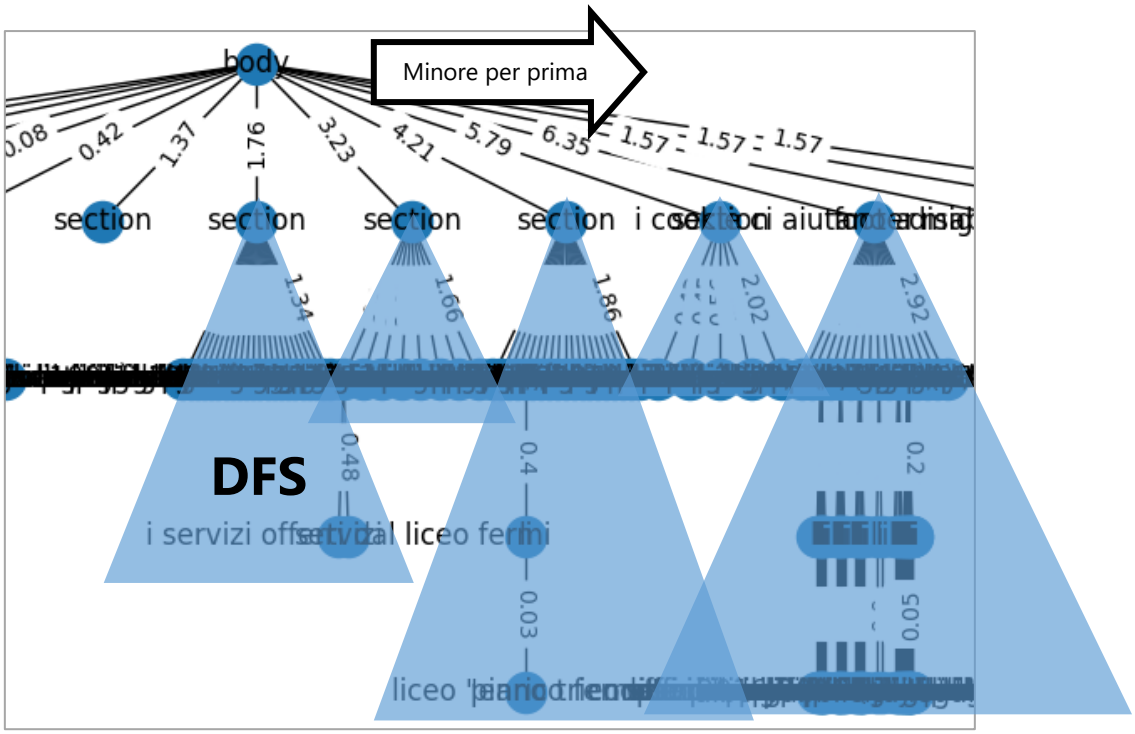


Figura 7. Metodo ***`plot()`*** *chiamato sul NDOM di https://www.liceofermicanosa.edu.it/ e illustrazione del funzionamento dell'algoritmo di ricerca.*

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 $O(|Nodi|) = O(b^h)$.
- Il percorso di un nodo obiettivo non è necessariamente quello dal costo minimo: una persona potrebbe trovare una sezione di suo interesse esaminando una parte centrale dell'intera pagina (e perdendo molto tempo) quando questa stessa sezione può essere contenuta chiaramente nel footer.

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.

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

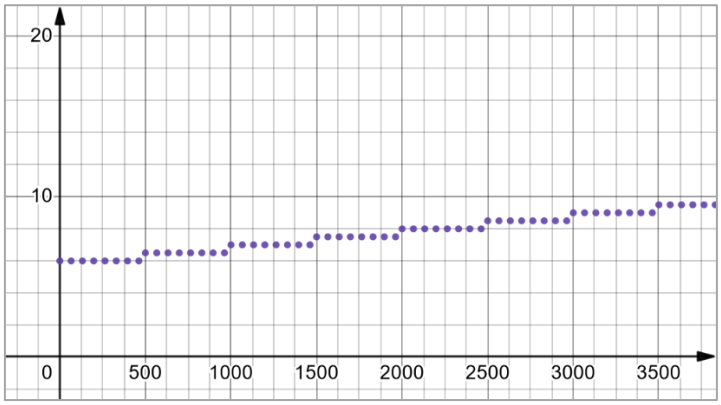


Figura 8. Grafico funzione $costoDefaultTask$.
https://www.desmos.com/calculator/2epakrbyrj

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 1600 e 900 (la mia risoluzione schermo).

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.

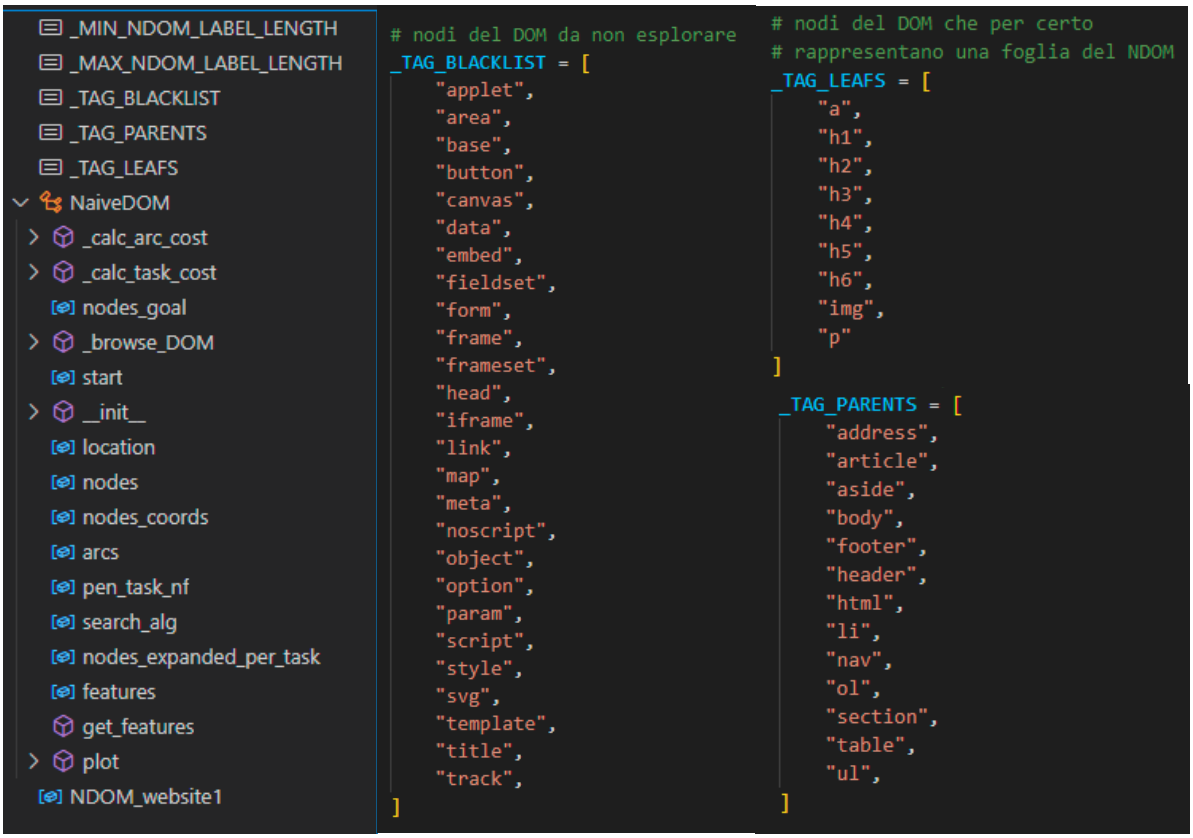


Figura 9.

Il metodo **`_browse_DOM`** è quello adibito alla costruzione del NDOM. 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.

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`
`/agent/ndom/benchmark/benchmark_full.xlsx`

In questa sezione valutiamo l'algoritmo di ricerca costruito mettendolo a confronto con altri algoritmi non informati DFS, BFS e LCFS. Consideriamo tutti gli indirizzi web rappresentanti del DS. Costruiamo un NDOM per ciascun sito, e di volta in volta cambiamo algoritmo di ricerca dei nodi obiettivo.

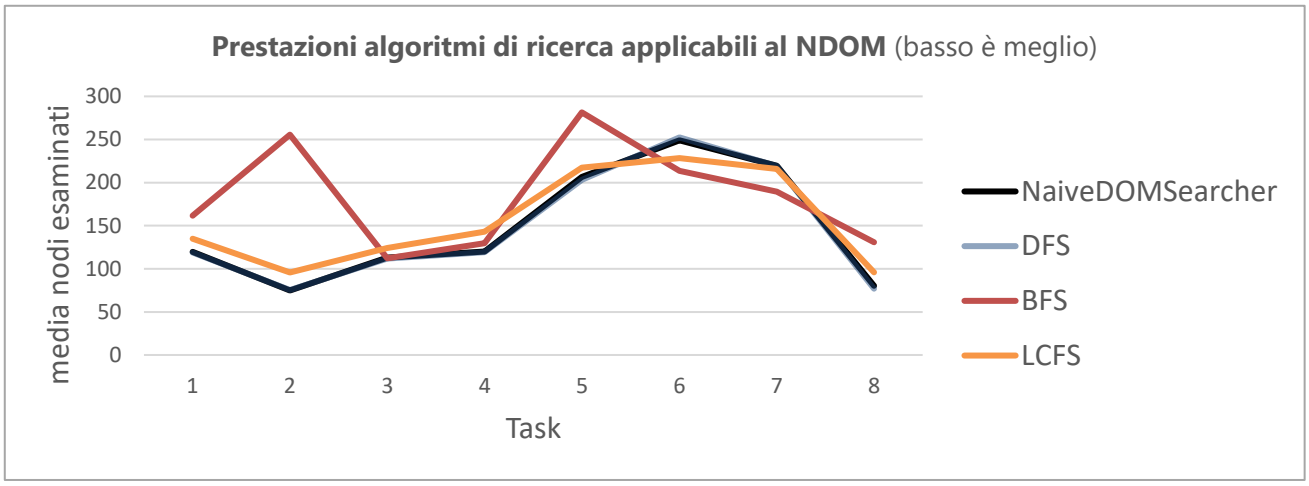


Figura 10.

Il grafico mostra come per il Task 3 (Notizie), Task 4 (Progetti) e Task 8 (Contatti) 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 in questo caso un nodo obiettivo tende ad essere posizionato 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 (in media 26 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`** (task di regressione). Approcci usati: classico; case-based; con metodi Ensemble.

Strumenti utilizzati e Decisioni di progetto

Notebook: **`/agent/models/nb_supervised_learning.ipynb`**
oppure:
https://github.com/vodibe/icon-745751/blob/main/agent/models/nb_supervised_learning.ipynb

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

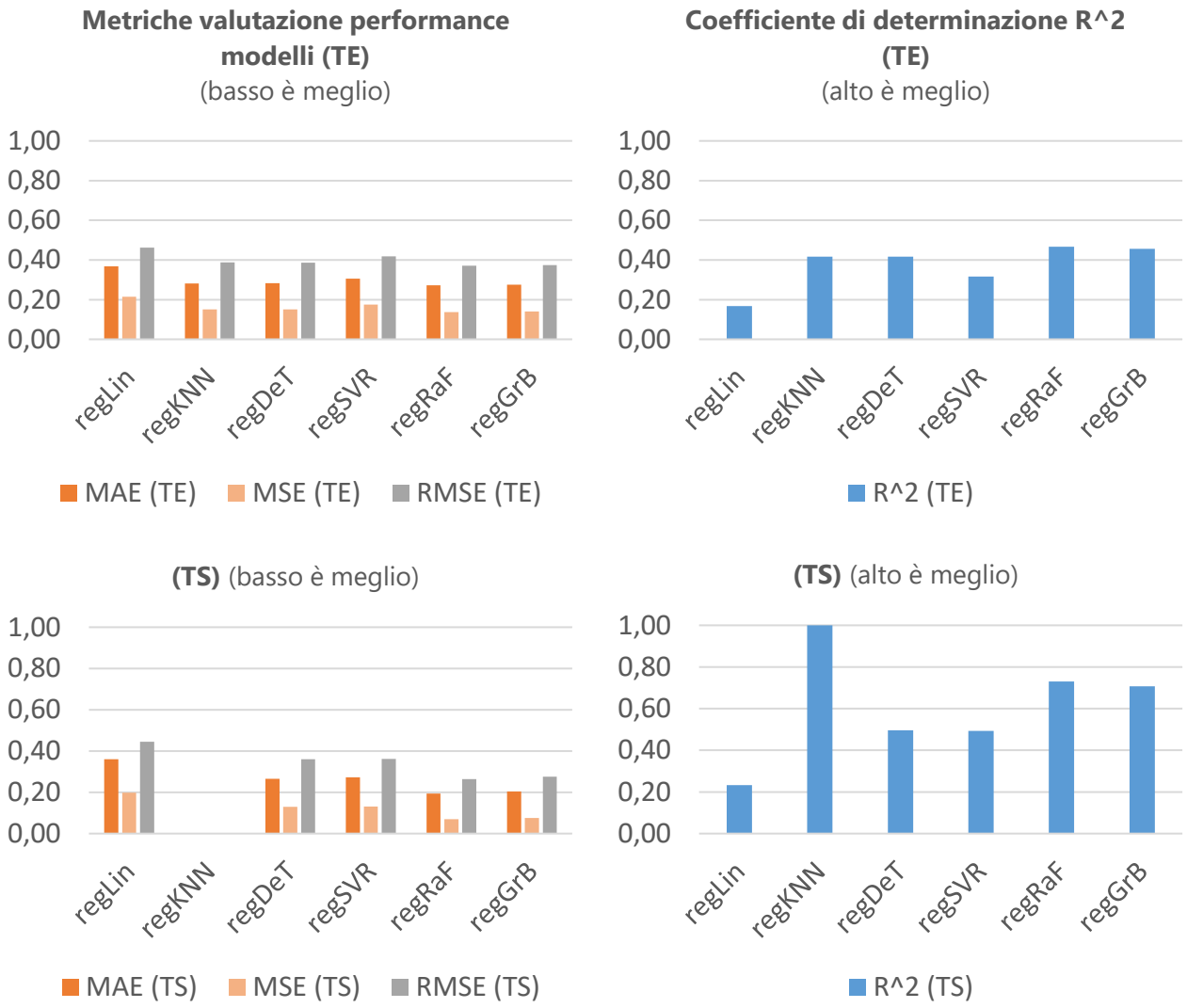
Valutazione

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

Per il task di regressione si sono calcolate diverse metriche, calcolate sui dati di training e di test.

- Mean Average Error (MAE): media dei valori assoluti tra le previsioni e i valori reali. A differenza del MSE, il MAE non penalizza tanto gli errori grandi.
- Mean Squared Error (MSE): media dei quadrati delle differenze, penalizza in modo più severo gli errori grandi rispetto a quelli piccoli.
- Root Mean Squared Error (RMSE): metrica comune poiché in molti casi le operazioni di radice quadrata sono più facili da gestire, soprattutto quando bisogna impiegare le derivate. Penalizza in modo molto elevato gli errori grandi.
- 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.

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



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, pur basandosi su un template, stravolge il suo assetto grafico (raro).
Se utilizziamo la configurazione:

- `K` pari a 12
- esempi simili ponderati su 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 dei cambiamenti della feature target.

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) è abbastanza buono. **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 notevole precisione. Entrambi 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 0,46. 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 0,46, questo è comunque un segno indicante un comportamento più "intelligente" rispetto a un modello baseline 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 ragionare con espressioni nel linguaggio di clausole di Horn (sottoinsieme della 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

Cartella: **`/agent/kb/`**

Nella KB scritta in Prolog, i fatti sono asseriti a partire dalle informazioni del dataset **`ds3_gt_final`**, cioè quello della sezione del SL. Le operazioni fatte sulla KB sono chiamate Jobs.

- **`kb_shared_facts.pl`** contiene i fatti a disposizione di tutti i Jobs;
- **`kb_shared_rules.pl`** contiene le regole a disposizione di tutti i Jobs;
- **`jobs/jobX_clauses.pl`** contiene fatti e regole utilizzabili solo per il Job X.

Un Job, tuttavia, potrebbe richiedere informazioni non presenti nel **`ds3_gt_final`**, ma sparse in altre KB. Ad esempio, il codice catastale del comune in cui si trova la scuola (associata alla Homepage) non è esplicitato in nessun fatto. Lo 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_kb`**. 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 (funzione **`query_wikidata_kb`**)

Decisioni di progetto

Interpretazione semantica

Individui: L'interpretazione del dominio è tale per cui gli individui sono:

- Pagina.
- Scuola (intesa allo stesso modo adottato dalla KB del MIUR, cioè ad es. liceo classico, istituto tecnico, liceo scientifico, …)
- Istituto scolastico che comprende 1 o più scuole.
- Posizione geografica.

Termini (Costanti, Variabili, Simboli di funzione): verranno elencati e spiegati successivamente assieme ai Job. Quello che verrà impiegato largamente da tutti i Job è il seguente.

```
schoolassoc(Url, School_ID)
```

Crea un'associazione semantica pagina-scuola.

Predicati generali:

```
page(schoolassoc(Url, School_ID),
        details(Width, Height, Load_time_ms, Template_ID, 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(Cod_Catastale), province(Prov_Name), region(Region_Name)).
```

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

Job 1

Output: `/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 raccogliere i contatti di tutte le scuole gestite da tale istituto. Può essere utile per avvertire simultaneamente i presidi delle scuole che condividono la stessa Homepage.

1) Criterio selezione 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 risultante da 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 criterio del redirect. Ecco perché nella query aggiungiamo la condizione `?S miur:CODICESCUOLA 'X'`.

Avremmo anche potuto ignorare questo vincolo e chiamare la seguente query, ma ciò avrebbe creato un numero elevato di fatti inutilizzati.

```
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

Output: `/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 selezione della pagina (Step 1), che ora si chiama `page_needs_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).
```

`is_good_template([1, 4, 5, 7]).` Questo fatto discende da Query BN 4-11, Query BN 12 (cap. successivo)

Job 3

Output: `/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) Criterio selezione pagina.

```
page_has_good_metric(schoolassoc(Url, School_ID)) :-
    page(schoolassoc(Url, School_ID), details(_, _, _, _, _, Ungrouped_multim), _, Metric),
    Metric >= 3.7,
    Ungrouped_multim =< 9.
```

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. Per generalizzare, chiamiamo Place un'area geografica da esaminare. Anche una regione è un esempio di Place. Quest'area ha una certa frequenza relativa quando essa 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 al punto 2) 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 come 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**. 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),

    findall(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

Output: `/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. Ad es. la provincia di Barletta-Andria-Trani verrà ignorata in quanto la capitale (in questo caso ne abbiamo addirittura 3) non è "Bari". La strategia che seguiremo è la seguente.

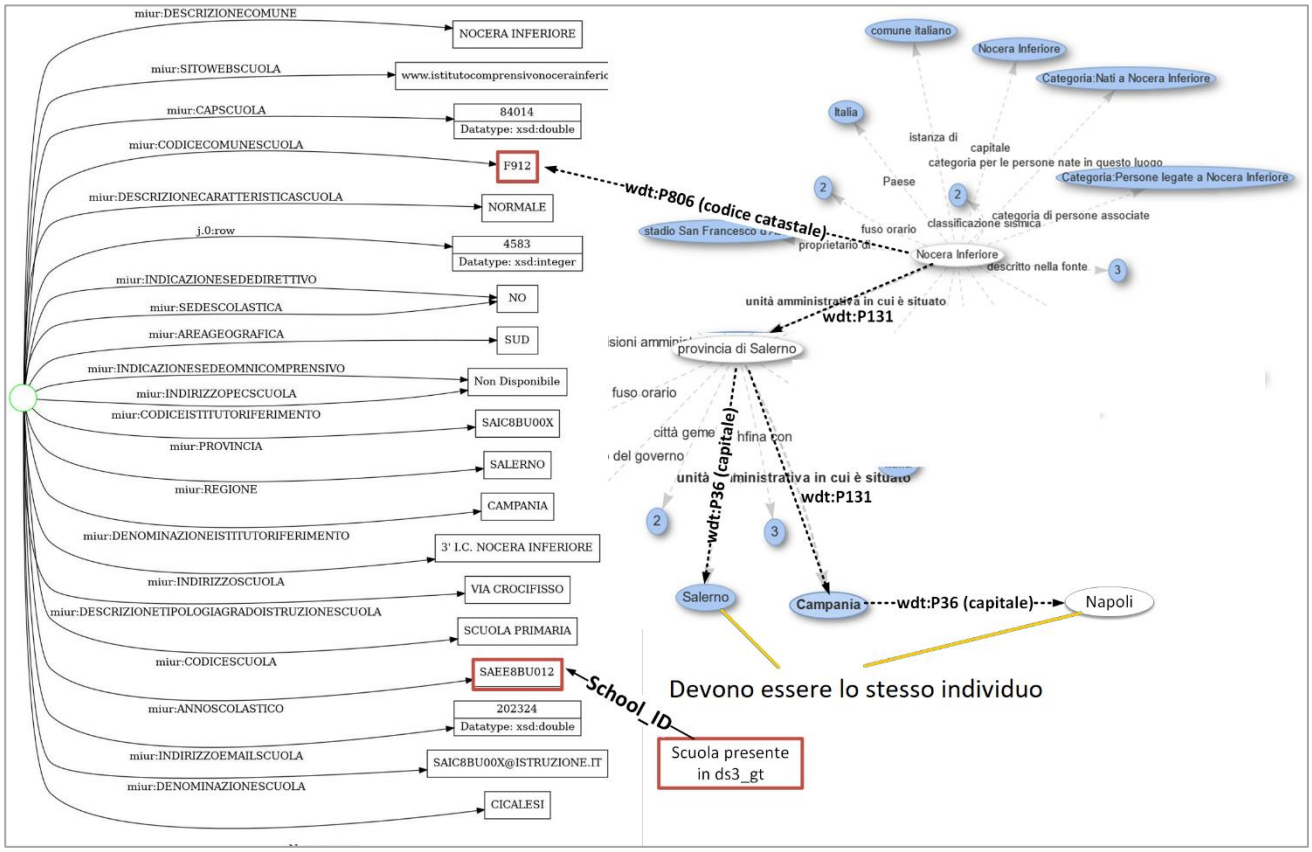


Figura 11. Navigazione nel web semantico per il Job 4.

1) Criterio selezione pagina: uguale a quello 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**, in questo Job, si compone di 2 argomenti, l'ID della scuola e il simbolo di funzione **province(Province_Name)**. Il nome esteso 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 e sono entrambe delle DataType Property.

Si è deciso di non impiegare il simbolo di funzione **province(Province_Code)** in quanto su Wikidata ci sono delle province che non dispongono di questa informazione. Un esempio è la città di Pordenone che ad oggi rientra nell' Ente di decentramento regionale di Pordenone. In teoria il codice è TN, ma non esiste una proprietà che assume questo valore.

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
```

Figura 12. Query visualizzabile a questo link. Nessun risultato, quindi Pordenone è una città localizzata in una provincia diversa dalla provincia del capoluogo di regione (Trieste).

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.

Job 5

Output: `/agent/kb/jobs/job5_output.txt`

Obiettivo: Individuare, per ogni regione italiana, i template più popolari.

1) Tupla RTC: Regione, Template_ID, Count.

```
is_rtc_tuple(rtc(Region, Template_ID, Count)) :-
   findall(
      School_ID,
      (school_is_in_place(School_ID, Region),
       page(schoolassoc(_, School_ID), details(_, _, _, Template_ID, _, _), _, _)
       ), L
   ),
   is_list_length(L, Count).
```

2) Raggruppamento tuple RTC per Regione. Ora dovremo asserire dei simboli di funzione rtc_grouped(Region, [tc(1, x), tc(2, y), ...]), ad indicare che per la regione Region ci sono x scuole che usano il template #1 e così via.

```
rtc_tuples_grouped_by_region(List_RTC_Tuples, Region, List_Templates,
RTC_grouped_by_region) :-
   findall( tc(Template_ID, Count),
         (member(Template_ID, List_Templates),is_rtc_tuple(rtc(Region, Template_ID, Count))),
         TC_Rank_Unordered
```

```
        ),
    predsort(tc_order, TC_Rank_Unordered, TC_Rank_Ascendant),
    reverse(TC_Rank_Ascendant, TC_Rank),
    RTC_grouped_by_region = rtc_grouped(Region, TC_Rank).
```

Job 6

Output: `/agent/kb/jobs/job6_output.txt`

Obiettivo: Individuare gli URL delle pagine aventi un punteggio tra i migliori 3 registrati.

1) Ricerca dei migliori 3 punteggi registrati.

```
are_best_metrics(Best_Metrics, Top) :-
    findall(Metric, page(_, _, _, Metric), Metrics),
    setof(M, member(M, Metrics), Metrics_WO_Dups),
    sort(0, @>=, Metrics_WO_Dups,  Metrics_Desc),
    findall(X, (nth1(I, Metrics_Desc, X), I =< Top), Best_Metrics).
```

2) Ricerca delle pagine.

```
are_best_pages_url(List_Best_Pages_Url, Top) :-
    are_best_metrics(Best_Metrics, Top),
    findall(Url,
            (page(schoolassoc(Url, _), _, _, Metric), member(Metric, Best_Metrics)),
             List_Best_Pages_Url
            ).
```

Osservazione su LOD

In questa fase si è potuto constatare come il dataset delle scuole offerto dal MIUR in formato CSV e RDF sia un esempio di LOD a 4 stelle: libera consultazione + formato strutturato e leggibile dall'interprete Python + formato open source + identificazione degli elementi delle triple mediante URI. Nel Job 4 si è contestualizzato il significato di alcune proprietà, collegandolo con altre KB e trasformando di fatto il dataset in un LOD a 5 stelle.

Rete Bayesiana

Sommario

Nella fase iniziale di costruzione del Ground Truth, è stato possibile visionare i template ad oggi in commercio e anche tutte le Homepage; pertanto si è potuta intuire l'esistenza di dipendenze tra le varie features. In questo progetto sfruttiamo i PGM per ideare delle fasce di classificazione dei template.

Strumenti utilizzati

Il modello grafico usato è una Rete Bayesiana per la quale si è imposto il vincolo di discretizzare tutte le variabili che fossero continue, in quanto la libreria Python pgmpy supporta l'apprendimento e l'inferenza probabilistica solo su questo tipo di BN [5]. Il mapping del dataset di partenza `ds3_gt` è descritto nel dizionario `DS_DISCRETE_MAPPING_DEFAULT`.

Si è deciso di usare il motore di inferenza esatta Variable Elimination, vista la BN non troppo complessa.

Decisioni di progetto

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

Struttura (costruita, non appresa)

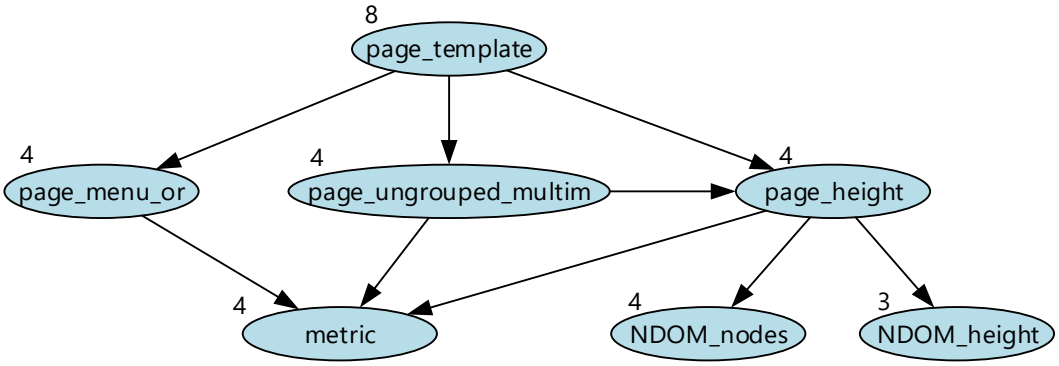


Figura 13. BN + cardinalità variabili.

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 14).

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 e numero di elementi confusionari. In teoria l'arco tra `page_height` e `metric` non dovrebbe esistere (come spiegato nella sezione *Costruzione del Ground Truth*), però ora `page_height` è discreta, e l'utente può facilmente rendersi conto se una pagina è breve (assumiamo che sia <=2600px), media (<=5200px), lunga (<=7200px) e troppo lunga (>7200px).

Venendo alle variabili **NDOM_nodes** e **NDOM_height**, entrambe sono indipendenti data l'altezza della pagina (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 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.

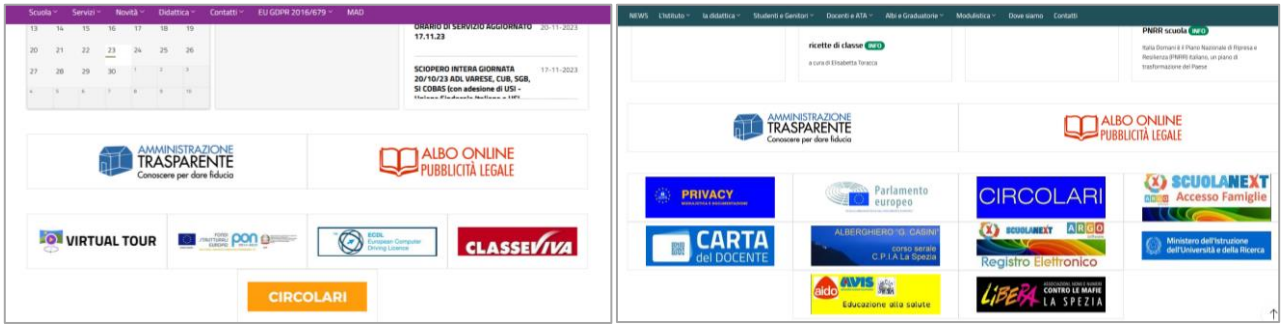


Figura 14. https://www.patettacairo.edu.it/ e https://www.alberghierolaspezia.edu.it/

Apprendimento parametri

CPTs: **/agent/pgm/bif/bn_MLE.bif**
/agent/pgm/bif/bn_MAP.bif

Un parametro è una cella di una CPT, cioè la probabilità che una variabile aleatoria assuma un determinato valore del suo dominio 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.

Sono stati impiegati due approcci di apprendimento:

Stimatore MLE. Fattibile in quanto sappiamo per certo che il dataset **ds3_gt** è pienamente rappresentativo dell'ambito trattato nel progetto: le uniche regioni di cui non disponiamo informazioni sono tali per cui neanche il MIUR ne è a conoscenza. I parametri stimati sono tali da massimizzare la loro likelihood, conosciuto il dataset.

$$\vec{\boldsymbol{\theta}}^* = \arg\max_{\vec{\boldsymbol{\theta}}} P(DS|\vec{\boldsymbol{\theta}})$$

Data la CPT del nodo X_i e data l'assegnazione $pa(X_i)$ delle variabili genitore, la stima del parametro è:

$$P\big(X_i = x_j \,\big|\, pa(X_i)\big) = \theta^*_{x_j|pa(X_i)} = \frac{\#\left(x_j \wedge pa(X_i)\right)}{\#\big(pa(X_i)\big)}$$

Stimatore Bayesiano (MAP). Fattibile in quanto l'osservazione di ogni singolo template (Figura 2) fa intuire delle probabilità a priori sotto forma di pseudo contatori **BN_MAP_PRIORS** di un parametro. Questo stimatore risolve 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.

Dato un nodo X_i con $dom(X_i) = \{x_1, \dots, k_K\},\ K > 2$, la sua CPT si comporrà di tante righe (a seconda delle possibili combinazioni di valori dei nodi genitore) e K colonne, che formano K parametri. Per una specifica riga, la distribuzione di probabilità è descritta da una variabile aleatoria avente distribuzione pari a quella di una $Dirichlet$.

$$P(\overrightarrow{\boldsymbol{\theta}_{\boldsymbol{X_i|pa(X_i)}}}|DS) = Dirichlet_{\alpha_{x_1|pa(X_i)},\dots,\alpha_{x_K|pa(X_i)}}\left(\underbrace{\theta_{X_i=x_1|pa(X_i)}, \dots, \theta_{X_i=x_K|pa(X_i)}}_{somma=1}\right)$$

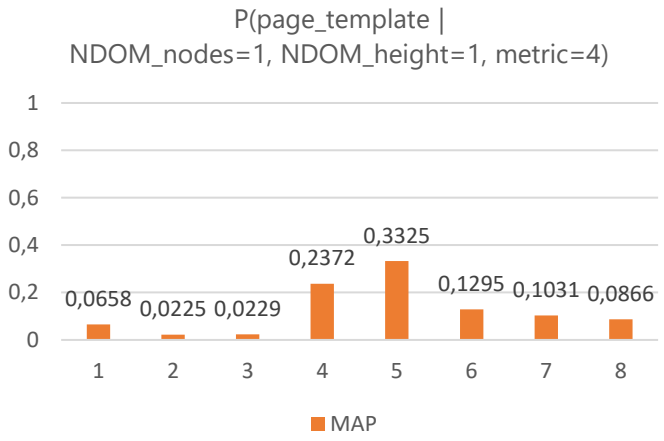
Calcolando la media di questa variabile aleatoria per l'i-esimo parametro, possiamo avere una stima.

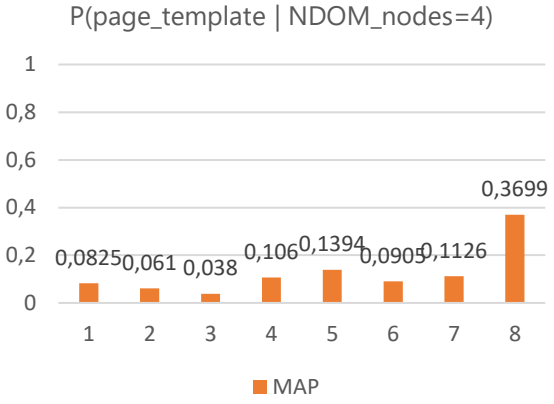
$$P(X_i = x_j \mid pa(X_i)) = \theta^*_{x_j|pa(X_i)} = \frac{\#\left(x_j \wedge pa(X_i)\right) + \alpha_{x_j|pa(X_i)}}{\#\left(pa(X_i)\right) + \sum_m \alpha_{x_m|pa(X_i)}}$$

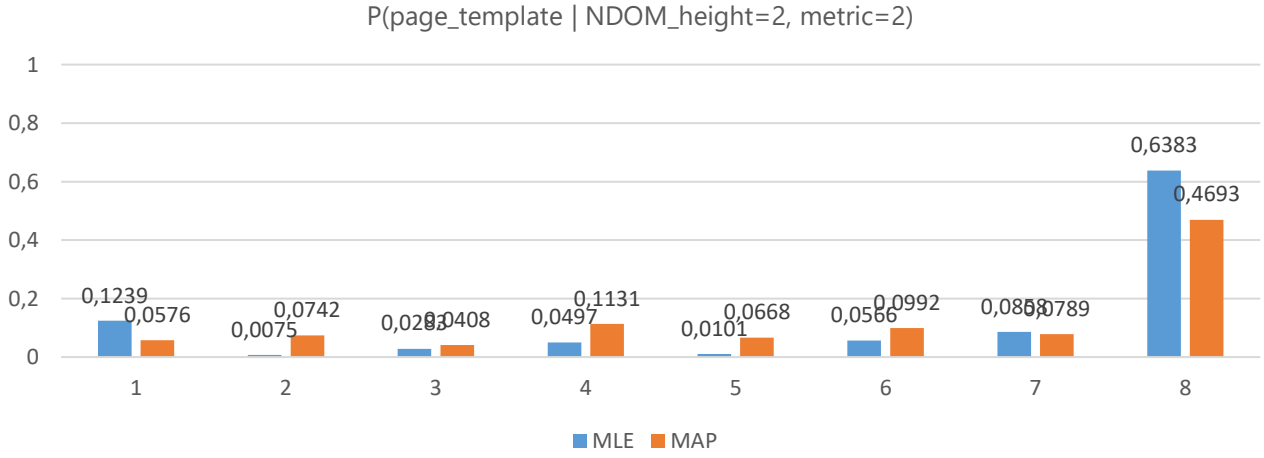
Query BN 1-3

Output: **/agent/pgm/bif/bn_MLE_query.txt**
/agent/pgm/bif/bn_MAP_query.txt
/agent/pgm/bif/charts.xlsx

Obiettivo: Data una pagina per la quale conosciamo la metrica e la struttura del NDOM, qual è la probabilità che sia basata su determinato template.





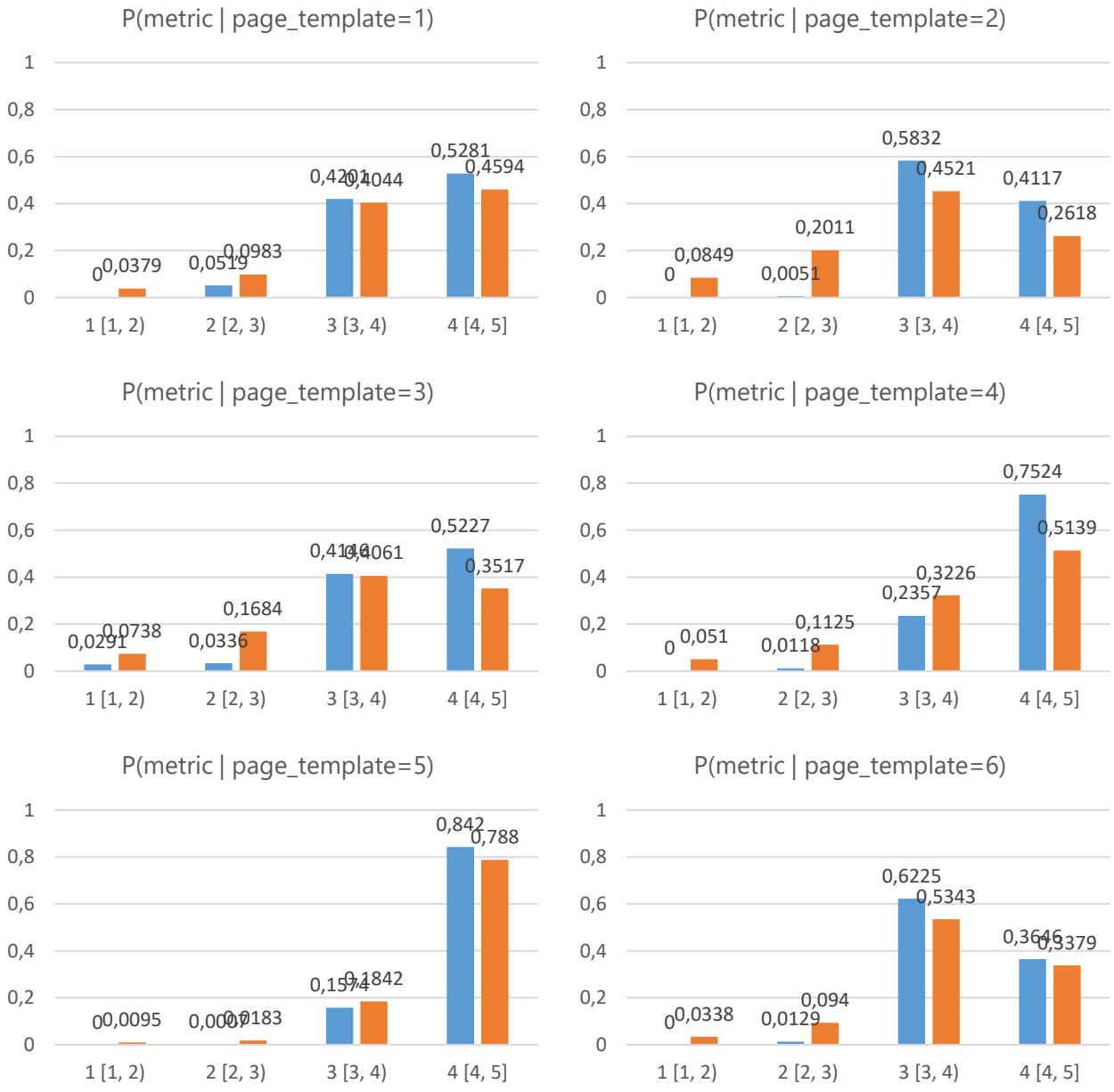


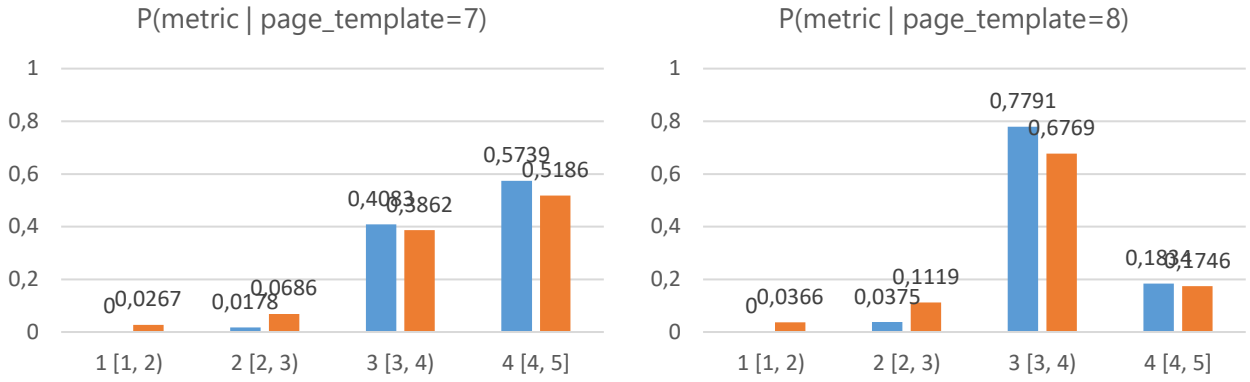
Nel primo grafico si è considerata una pagina "modello", cioè avente ottima struttura NDOM (<=500 nodi, <5 altezza) e un'ottima fascia di punteggio ([4, 5]). Emergono in positivo 2 template (#4 e #5) che rendono plausibile il fatto che una tale pagina sia basata su di essi.

Dagli altri due grafici emerge in negativo il template #8, principale responsabile di una struttura del NDOM non ottimizzata. Nel grafico in basso assumiamo un'altezza esagerata e un punteggio basso ([2, 3)): il template #8 è responsabile con il 46% di probabilità.

Query BN 4-11

Obiettivo: Data una pagina per la quale conosciamo solo il suo template, calcolare la probabilità di avere una determinata fascia di punteggio.



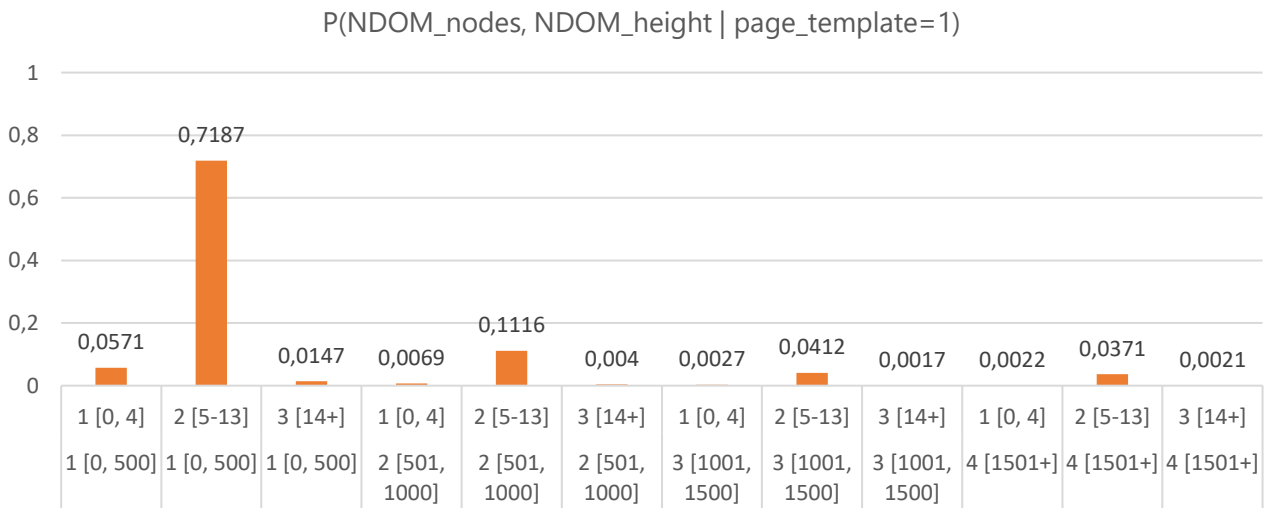


In questi 8 grafici si osserva la situazione per gli altri template. Per prima cosa individuiamo i template per i quali la probabilità che la fascia di metrica [4, 5] è superiore al 50%. Ritroviamo il #4 e #5, ma anche il #7. Il template #1 non è comunque da escludere (46%) e lo approfondiamo nella Query 12.

Successivamente individuiamo i template di fascia inferiore, che hanno una probabilità alta di ottenere una metrica compresa in [3, 4) e allo stesso tempo una probabilità irrisoria (<20%) di ottenere un punteggio < 3. Rientrano i template #3, #6 e #8. In ultima fascia rientra il template #2.

Query BN 12

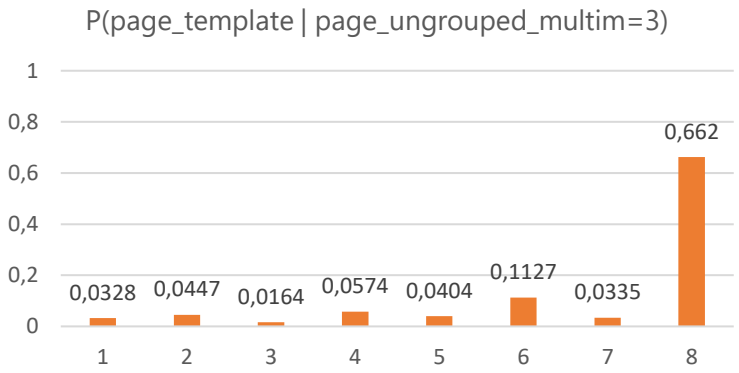
Obiettivo: Probabilità che una pagina con template #1 disponga di una determinata struttura del NDOM.



Emerge che nel 71% dei casi, una pagina che segue questo template è anche tale da avere una struttura del NDOM che si ritiene buona. Questo risultato, assieme a quello della Query 4, ci permette di inserire il #1 in fascia alta, assieme a #4, #5 e #7.

Query BN 13

Obiettivo: Probabilità che una pagina avente un numero di elementi caotici tra 11 e 20 segua un determinato template.



Il #8 è un template particolare, che come visto nelle query precedenti, rende assai probabile l'ottenimento di un punteggio [3, 4); e questo è spiegato dal grafico qui mostrato. Il template non permette una strutturazione del contenuto al di fuori del menu, e quindi costringe l'inserimento dei link aggiuntivi della pagina in sezioni non catalogate.

Ontologia di dominio

Sommario

Nella sezione "*Ragionamento relazionale, Web semantico*" si è potuto consultare l'endpoint SPARQL della KB remota, ma non è stato possibile trovare in rete l'ontologia formale `.owl` (o `.rdf`) delle scuole. In questa sezione ne costruiamo una che governa i concetti scolastici del paese Italia e la si impiega per ricostruire alcune righe del dataset **`ds1`**.

Strumenti utilizzati

Ontologie importate: **`/agent/ontology/imported/`**

Ad oggi esiste una rete di ontologie usate dalla PA italiana (daf-ontologie-vocabolari-controllati) che ci può essere d'aiuto, per cui invece che creare da zero una nuova, si impiegano i concetti di Luogo, Indirizzo (ed altri) già assiomatizzati nell'ontologia **CLV (Indirizzi/Luoghi Core Location Vocabulary)**. Innanzitutto, si è dato un'occhiata alla documentazione.

- ontologia in più notazioni. Si è scelta quella RDF/XML (`CLV-AP_IT.rdf`).
- visualizzazione in HTML con LODE (solo inglese) o LodView (inglese e italiano)
- visualizzazione a grafo con WebVOWL

Le annotazioni di alcune classi ci suggeriscono che all'occorrenza, quando è necessario definire individui riconosciuti a livello nazionale (ad es. le aree geografiche Nord, Sud, Centro, Isole) si importeranno i dizionari controllati (**`ontology/imported/controlled_dicts/`**).

Osserviamo inoltre che l'ontologia CLV importa a sua volta un'ontologia più astratta e imprescindibile: l'ontologia **L0 (Top-Level)**. Sarà utile in quanto una Scuola può considerarsi un particolare tipo di Agente. Anche qui si è dato prima un'occhiata alla documentazione:

- visualizzazione in HTML con LODE (rimanda alle altre due visualizzazioni)

La creazione dell'ontologia avviene con il software protégé e i seguenti Plug-in:

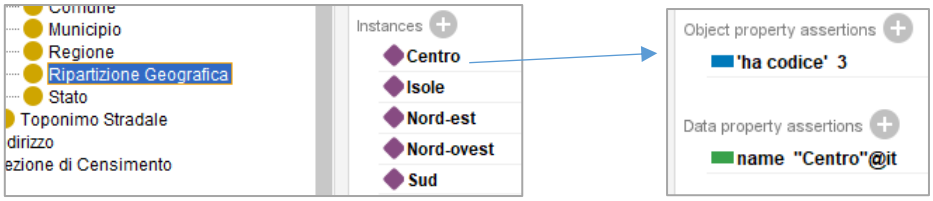
- Pellet (reasoner) v.2.2.0
- SWRLTab Plugin v2.0.11
- snap-sparql-query-plugin v6.0.0

Decisioni di progetto

Ontologia: **`/agent/ontology/ambitoscuola_v1.owl`**

Osservazioni e assunzioni iniziali

Innanzitutto, notiamo che importando il dizionario controllato **`geographical-distribution.rdf`**, disponiamo ora di 5 nuovi individui elementi della classe Ripartizione Geografica (definita in CLV), i cui nomi sono esattamente i possibili valori della feature **`AREAGEOGRAFICA`** del dataset **`ds1`**.



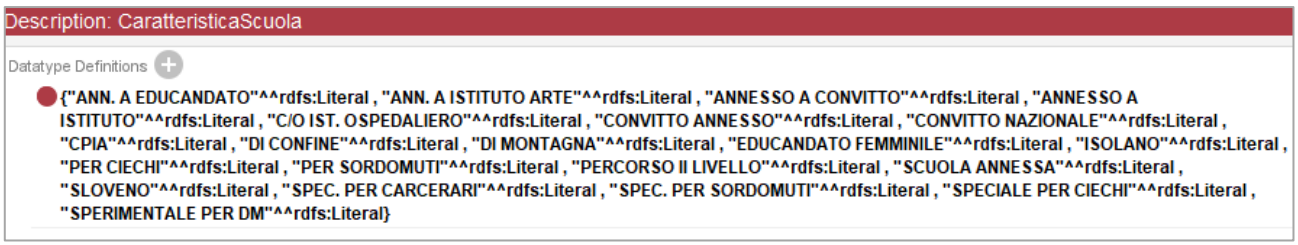
Non si fa la stessa cosa per i dizionari delle regioni, province e città italiane, perché se ne impiegheranno solo alcune.

Le scelte progettuali dell'ontologia sono le seguenti, e sono state formulate visitando la pagina di descrizione del dataset (sezione Tracciato record):

1. Per "Scuola" noi intendiamo un plesso con uno specifico Grado Istruzione: "scuola infanzia", "scuola primaria", "scuola media", "liceo classico", "istituto tecnico industriale", …
2. Ciascuna scuola presente nel dataset è tale per cui:
   2.1. esiste/svolge gli insegnamenti nell'anno scolastico 2023/2024.
   2.2. ha una serie di informazioni relative a: codice e nome; la sua ubicazione.
   2.3. dispone anche della proprietà "Caratteristica Scuola".
   2.4. conosce la scuola a capo di un eventuale istituto omnicomprensivo correlato ad essa.
   2.5. fa capo a un Istituto Scolastico, che assume codice e nome pari al codice e nome della scuola che si considera "Sede Direttivo".
   2.6. dispone di 0 o più metodi per contattarla;
   dispone inoltre della proprietà "Sede Scolastica". Quest'ultima risulta avere valore "NO" in tutte le righe del database e non si è riuscito a capire in base a cosa dipende il suo valore, perché non esplicitata nel Tracciato record.

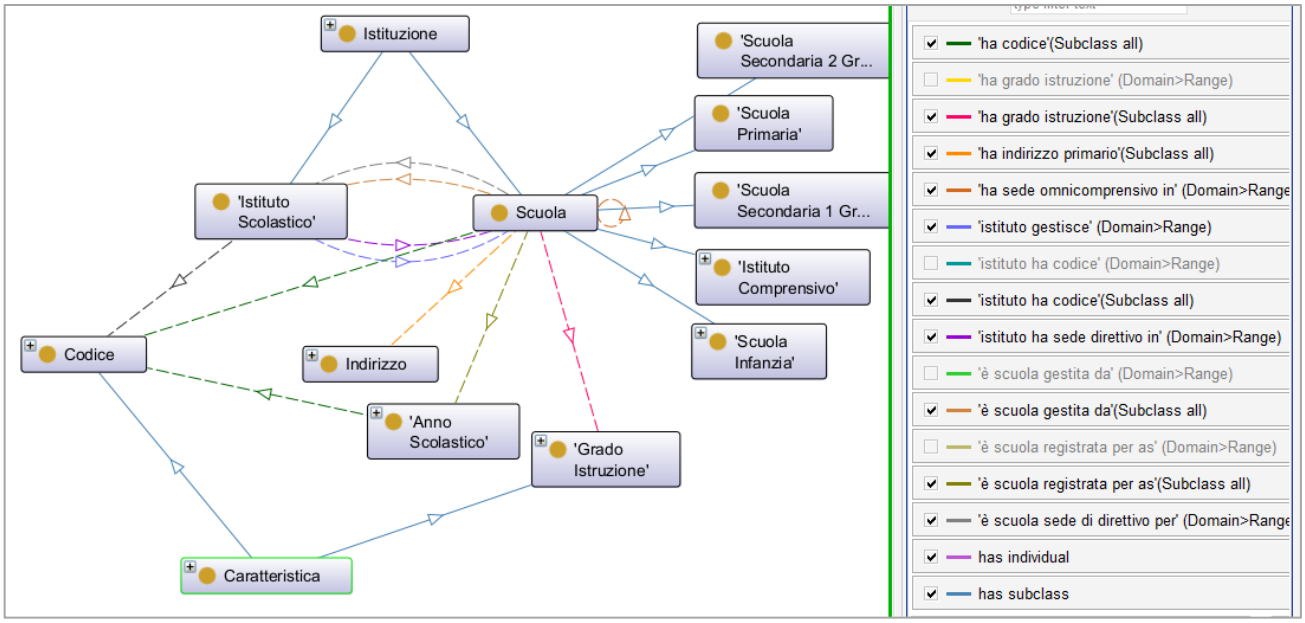
Datatypes

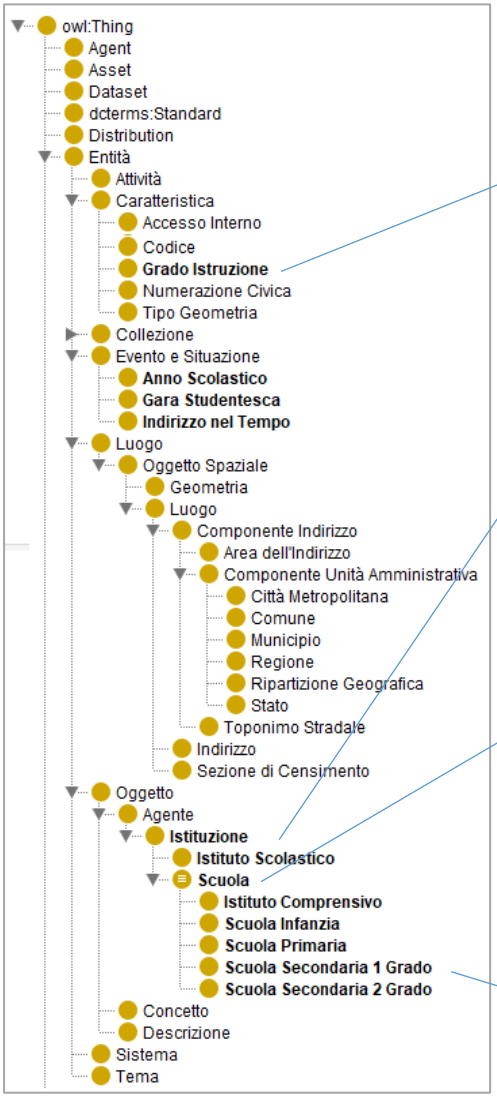
Per il punto 2.3 si è creato un Datatype apposito perché si è deciso di trattare questa caratteristica come una DataType Property.

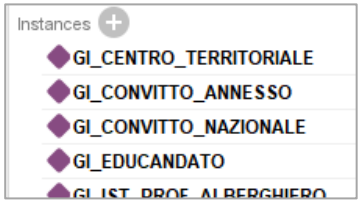


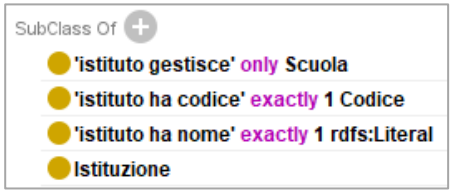
Classi e ObjectType Properties

Le assunzioni 1, 2.1, 2.2 e 2.4 vengono implementate con questo grafo semantico delle classi.





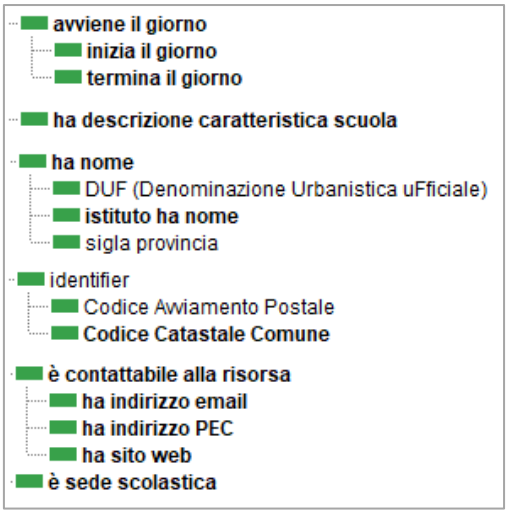




Tra le ObjectType Properties indicate nel grafo, le seguenti 4 sono state progettate come segue. Tutte queste, tranne "istituto gestisce" sono funzionali, vale a dire che ogni soggetto ha al più un oggetto: $\forall x,y,z\ (\ p(x,y) \wedge p(x,z) \rightarrow y = z\ )$. E' necessario specificarla perché il linguaggio OWL non assume la UNA. Questa decisione è comunque "rafforzata" dal fatto che si è asserito che una Scuola sia un individuo tale per cui 'è scuola gestita da' exactly 1 'Istituto Scolastico' .

DataType Properties

Si sono aggiunte queste proprietà, utili a implementare l'assunzione 2.6 e 2.3.



Regole SWRL

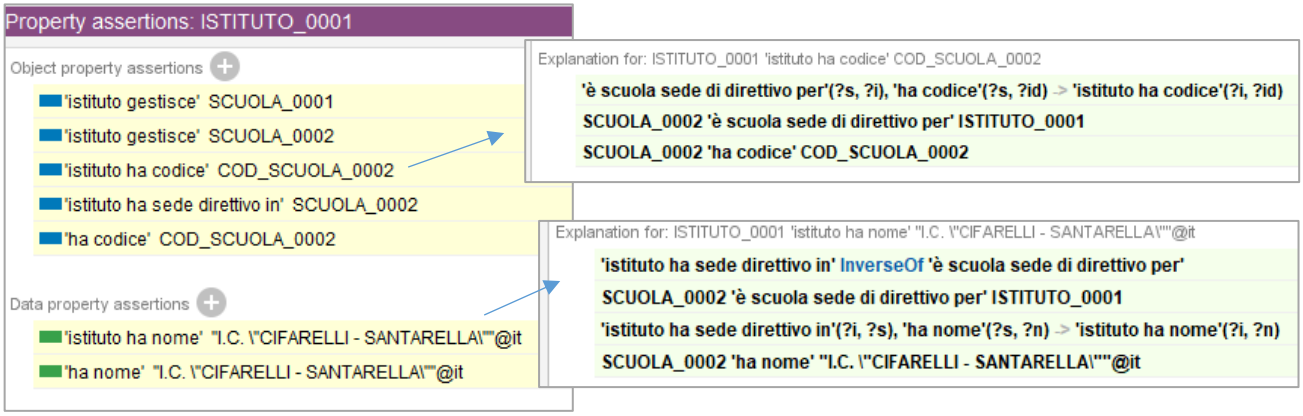
L'assunzione 2.5 può essere implementata in due modi: (a) cercando delle asserzioni da inserire nella TBox usando gli operatori della DL; (b) con le seguenti due regole SWRL.

Il linguaggio SWRL [6] sfrutta i costrutti del linguaggio OWL + dei costrutti sintattici per creare delle clausole di Horn da cui poter dedurre conoscenza. Con le regole SWLR possiamo dedurre nuove asserzioni della ABox che con le DL potrebbero non esistere. Ovviamente ciò richiede un reasoner in grado di analizzare queste clausole, ad es. Pellet.

o:ScuolaSedeDirettivoPer(?s, ?i) ^ CLV:hasIdentifier(?s, ?id) -> o:IstitutoHaCodice(?i, ?id)

o:IstitutoHaSedeDirettivoIn(?i, ?s) ^ l0:name(?s, ?n) -> o:IstitutoHaNome(?i, ?n)

Entrambe le regole devono essere eseguite dopo l'avvio del reasoner, in quanto per un individuo, una proprietà può essere dedotta e non necessariamente esplicitata da un assioma. Selezionando un istituto di esempio e avviando il reasoner otteniamo queste due spiegazioni.



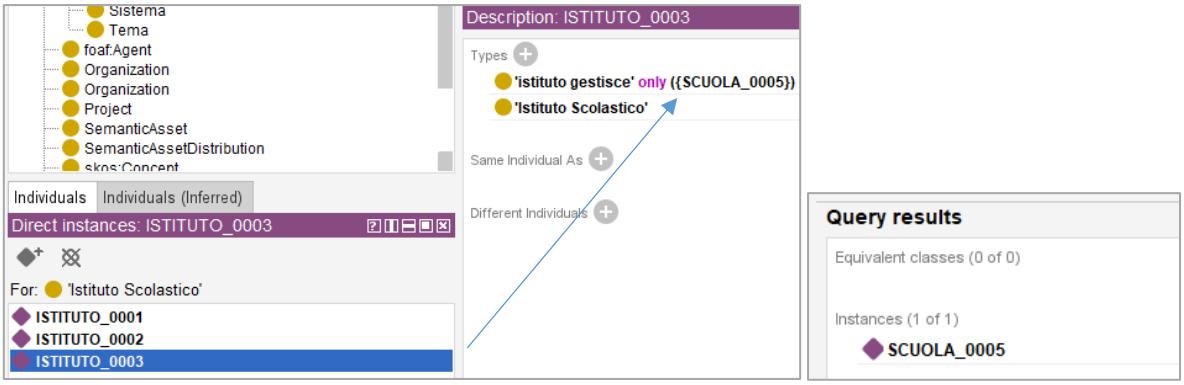
Query DL 1

Query: `/agent/ontology/query/query_dl1.txt`
`/agent/ontology/query/query_dl2.txt`

Obiettivo: Formulare la classe delle scuole "autonome", cioè tali per cui esiste un istituto che gestisce solo quella scuola.

Scuola and 'è scuola gestita da' exactly 1 ('Istituto Scolastico' and 'istituto gestisce' exactly 1 Scuola)

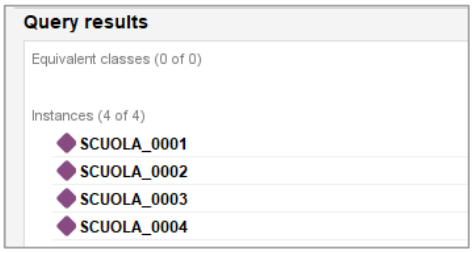
Da notare che se, ad es., avessimo specificato soltanto che l'istituto `ISTITUTO_0003` gestisce `SCUOLA_0005` (SENZA esplicitare l'unicità) questa query non funziona. Questo perché OWL adotta l'assunzione del mondo aperto. Quindi, dichiariamo il seguente assioma.



Query DL 2

Obiettivo: Formulare la classe delle scuole del Centro, Sud Italia e Isole che dispongono di almeno una risorsa (email, PEC o sito) per essere contattata. Escludere le scuole superiori.

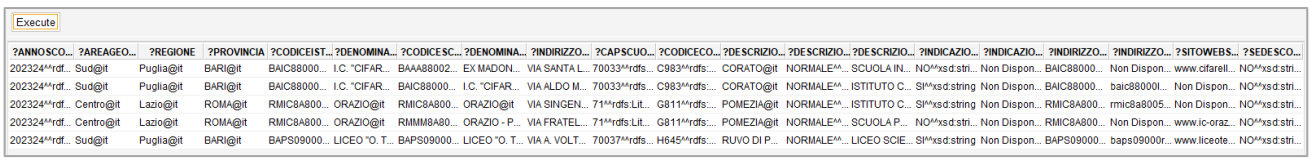
```
(not 'Scuola Secondaria 2 Grado')
and
'ha indirizzo primario' exactly 1 (
        'ha comune' some (
                'ha livello più alto' some (
                        'ha livello più alto' some (
                                'ha ripartizione geografica' exactly 1 (
                                        'ha nome' some (not{"Nord-est"@it, "Nord-ovest"@it}))
                        )
                )
        )
)
and
('è contattabile alla risorsa' min 1 (xsd:anyURI))
```



Query SPARQL 1

Query: `/agent/ontology/query/query_sparql1.rq`
`/agent/ontology/query/query_sparql1_output.txt`

Obiettivo: Estrarre informazioni degli individui della classe Scuola, presentandole nello stesso formato delle righe del dataset **ds1**.



I dati presentati con questa query SPARQL hanno stessa forma di quelli in tabella, e in più includono l'URI del DataType.

Conclusioni e sviluppi futuri

Si è partiti dall'argomento di rappresentazione della struttura delle pagine; questo è stato di aiuto per la fase successiva di apprendimento supervisionato. Per questioni di tempo non sono stati utilizzati strumenti che potessero ricavare automaticamente il valore della feature **`page_ungrouped_multim`**, il che può essere fatto ad esempio, usando tecniche di Webpage Segmentation (applicate sullo screenshot della pagina) che coinvolgono algoritmi di Clustering come DB-SCAN [7] , k-means o VIPS [8].

Successivamente, la KB trattata, oltre alle funzionalità di reporting (Job 1, 2), ci ha permesso di giungere alle conclusioni mostrate in Figura 17 (Job 5) e Figura 18 (Job 3 e Job 4).

Nelle regioni del centro-nord Italia il Gruppo Spaggiari Parma, titolare del template #8 "Prima Visione Web" (in rosso) gestisce in modo capillare l'infrastruttura web scolastica. Anche la Toscana non è esclusa: è il secondo template più diffuso. In sud Italia e nelle isole, gli studi di sviluppo stanno puntando sul template open source #5 (Bootstrap Italia). In Calabria e Sicilia è quello più impiegato, con un netto distacco su quello immediatamente più popolare (28-13 in Calabria; 41-28 in Sicilia); in Puglia, Basilicata e Sardegna è il secondo più impiegato. In Abruzzo e Campania c'è invece una forte diffusione del template #7 (Argoweb).

Ruolo importante ha avuto anche la Rete Bayesiana da cui è emerso che il template #8, seppur diffusissimo, non è il migliore in circolazione. Le scuole che lo adottano potrebbero valutare l'idea di virare su modelli open source che hanno dimostrato di essere più apprezzati.

Prima fascia	#1, #4, #5, #7
Seconda fascia	#3, #6, #8
Terza fascia	#2

Tabella 1.

Se si disponesse di informazioni di tutt'altro tipo rispetto a quelle trattate in questo progetto, quali costi di installazione, costi di manutenzione, preferenze dei presidi degli istituti, ecc… si potrebbe integrare l'argomento del CSP, in aiuto al problema di scelta del template.

L'ambito di questo progetto comunque, può essere esteso analizzando tutti i siti del dataset **`ds1`** e catalogando i risultati per grado istruzione di scuola (scuole elementari, medie, e superiori).

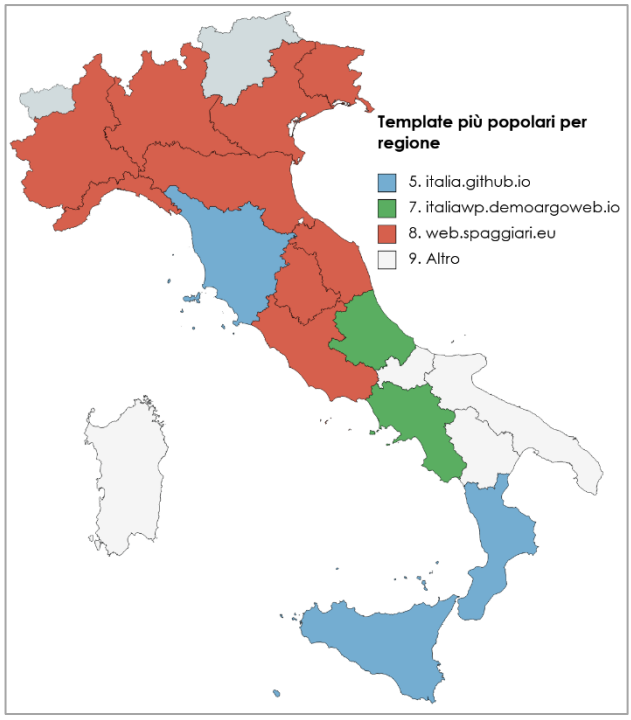


Figura 15. Escluse regioni a s.s. Trentino Alto Adige e Valle d'Aosta (le cui scuole non sono incluse in `ds1`*).*

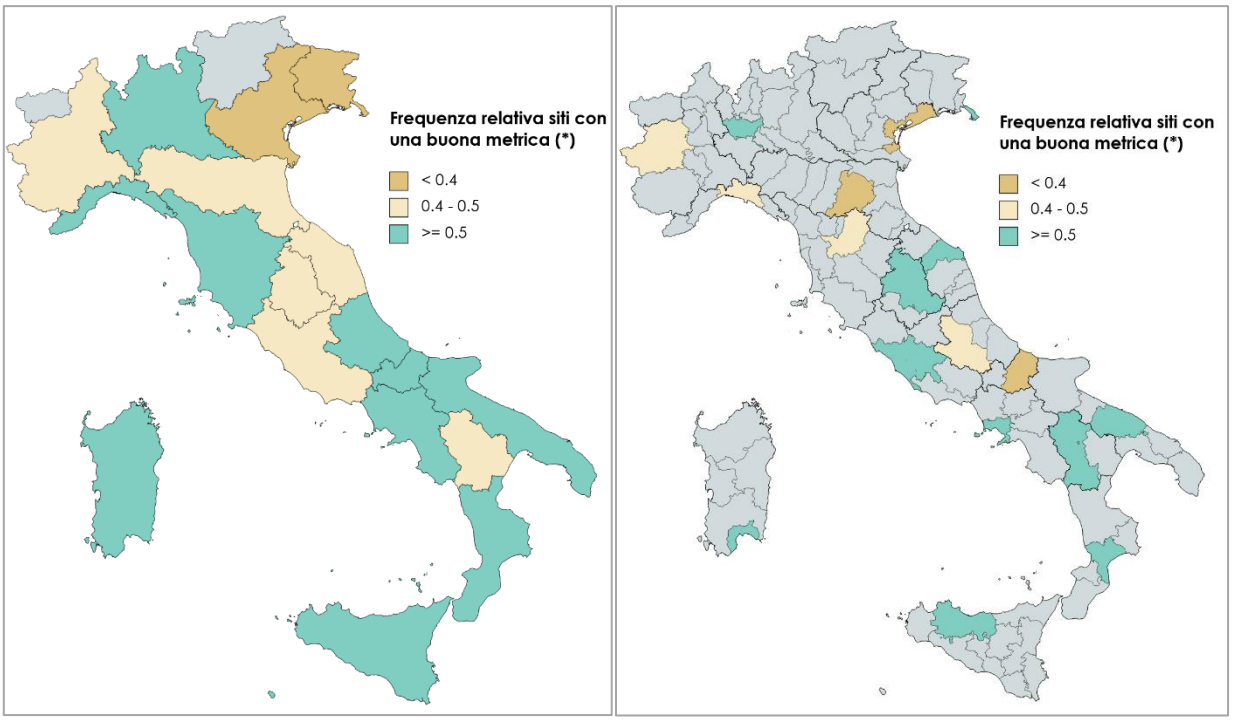


Figura 16. (*) Condizioni espresse nella clausola **page_has_good_metric**.

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/#supported-data-types.

[6] [Online]. Available: https://www.w3.org/submissions/SWRL/.

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

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