Linkage Sorgenti: Allineare Pagine Appartenenti a Sorgenti Differenti

Davide Massuda, Giulio Raponi, Marco D'Elia

16 Dicembre 2020

Abstract
Il web è uno spazio pieno di risorse da cui attingere una vasta quantità di informazioni. Sono presenti ormai su internet dati strutturati e non, di qualsiasi tipo. Negli ultimi anni è stato registrato un crescente interesse nell'utilizzo di dati sul web per molte applicazioni, ad esempio la ricerca sul web, la risposta alle domande e l'integrazione dei dati. In questo articolo presentiamo un metodo per l'allineamento di pagine web appartenenti a due sorgenti differenti. Sfruttando la ridondanza di informazioni presente nei diversi siti web siamo riusciti a sviluppare una tecnica non supervisionata in grado di ottenere ottimi risultati in termini di prestazioni. In particolare abbiamo osservato che, prendendo come riferimento per la valutazione più sorgenti relative al dominio del basketball, la soluzione proposta mostra valori medi di Precision: 92%, Recall: 97% e F1: 94%.

1 Introduzione

Ci sono molte sorgenti che forniscono informazioni sovrapposte tra loro, sia parlando di schemi e attributi, sia nelle istanze, ad esempio stesse entità in siti web differenti. Avere a che fare con sorgenti differenti equivale ad avere a che fare con molte eterogeneità nei dati, come ad esempio le unità di misura, le approssimazioni (granularità diverse) e le diverse rappresentazioni. La ridondanza delle informazioni è uno dei problemi più significativi che viene affrontato nel momento in cui dobbiamo integrare i dati, ma allo stesso tempo costituisce una delle opportunità che possiamo sfruttare. In questo articolo si tratterà di come, date in input due sorgenti, sia possibile trovare le corrispondenze tra le varie pagine delle due sorgenti stesse.

2 Descrizione del Problema

Siano X, Y due sorgenti definite come: X = $\{x_1, x_2, ..., x_i, ..., x_n\}$ e Y = $\{y_1, y_2, ..., y_j, ..., y_m\}$, dove i generici elementi x_i, y_j rappresentano un url della sorgente X ed Y rispettivamente.
Il problema è quello di determinare un insieme I = $\{(x_i, y_j),...\}$ tale che la generica coppia faccia riferimento alla stessa entità, $x_i \in X$, $y_j \in Y$.

3 Descrizione della Soluzione

In questa sezione descriviamo in modo più dettagliato come opera la nostra soluzione, confrontando più pagine relative a due siti web differenti. Come detto in precedenza assumiamo che le pagine siano già state collezionate, perciò non affronteremo il problema del crawling sul web.

3.1 Wrapping

Il primo task affrontato dall'algoritmo riguarda l'estrazione degli url delle varie pagine da dei file di testo. In particolare, il metodo utilizzato si occupa di estrarre gli url e aggiungerli ad una lista che verrà in seguito processata. Durante il riempimento di questa ci si è accertati che non ci fossero url duplicati.

```
loadSource(input: filename):
    url_list  //lista
    for url in filename:
        if url not in url_list:
            url_list <-- url
    return url_list
```

Il passo successivo è quello riguardante l'estrazione delle informazioni presenti nei siti web. Per ogni elemento precedentemente aggiunto in lista, attraverso l'utilizzo di una regola scritta nella notazione XPath: "//*[not(child::*)]/text()", siamo stati in grado di creare un'altra lista, contenente tutti e soli gli elementi foglia per ogni pagina web.

```
build_corpus(input: url_list):
    lista_corpus //list
    for corpus in url_list:
        lista_corpus <-- get_leaves(corpus)     // prendi tutte le foglie
    return lista_corpus
```

3.2 Tf-Idf e Adattamento al Contesto

Sulla falsa riga di ciò che viene fatto nell'Information Retrieval, in particolare nel suo impiego dell'algoritmo tf-idf, è stato possibile dar valore ad ogni parola estratta presente nelle diverse liste. Abbiamo considerato l'insieme delle liste come corpus e ogni singola lista come fosse un documento. Vengono quindi creati due corpus distinti, corrispondenti ai due siti web considerati. Questa operazione preliminare ci ha permesso di ricavare una tabella (Dataframe in Python) le cui righe fanno riferimento alle singole pagine web e le colonne invece sono relative ai vari termini presenti nel corpus. Dove il termine considerato non era presente nella specifica pagina web il valore corrispondente era 0; questo valore era più alto se presente in più documenti e ancora più significativo se presente in pochi documenti. In ogni corpus si è cercato di eliminare gli elementi i cui valori fossero uguali per tutte le pagine web o che fossero tutti positivi, filtrando così le colonne che presentavano solo i termini più rilevanti per le singole pagine.

```
annulla_term_com_TfIdf(input: dataframe):
    termini_rilevanti //list
    for colonna in dataframe:
        if not all_equal(dataframe[colonna]) and not_all_positive(dataframe[colonna]):
            // se i valori non sono tutti uguali e non sono tutti positivi
            termini_rilevanti <-- colonna
    return dataframe[termini_rilevanti]
```

Prima dell'applicazione di quest'ultimo metodo e, subito dopo, si è andato a lavorare sui termini comuni ai due insiemi; nello specifico sono state rimosse le colonne relative agli elementi che non erano presenti nei due corpus, permettendoci di lavorare su tabelle con dimensioni minori ottenendo così un processamento più rapido.

```
colonne_comuni(input: dataframe1, dataframe2):
    lista_colonne_comuni // lista
    lista_colonne_comuni <-- intersezione_colonne(dataframe1, dataframe2)
        // aggiungi in lista tutte le colonne comuni
    return dataframe1[lista_colonne_comuni], dataframe2[lista_colonne_comuni]
        // return i due dataframe con le sole colonne comuni
```

L'utilizzo del solo algoritmo tf-idf e del metodo "annulla_term_com_TfIdf" non basta però ad esaltare i termini discriminanti, in quanto il valore dei loro pesi era ancora troppo simile a termini comuni alle varie pagine e quindi meno rilevanti. Si è sviluppato un metodo che desse maggiore importanza a quei termini presenti in un unico documento all'interno del corpus. In particolare, il valore in questione è stato raddoppiato in modo tale da renderlo più incisivo rispetto agli altri.

```
increase_elem_value(input: dataframe):
    for colonna in dataframe:
    if soloUnValoreDiversoDaZero(colonna):
        valoreDiversoDaZero(colonna) = valoreDiversoDaZero(colonna)*2
        // se vi è un solo valore diverso da zero allora raddoppialo
    return dataframe
```

Il risultato alla fine di tutto questo processamento ha prodotto due Dataframe contenenti solo termini comuni e con valori dipendenti dalle singole pagine piuttosto che dal corpus stesso.

3.3 Allineamento delle Pagine

Una volta ottenuta una tabella determinante per il nostro dominio, l'ultimo task ha riguardato la necessità di confrontare le parole più importanti per ogni pagina all'interno del primo corpus con le stesse parole appartenenti al secondo. In particolare, se il valore di ogni singolo termine era maggiore di una soglia impostata, allora questo risultava rilevante per la pagina web. Questi termini venivano successivamente confrontati, per ogni pagina appartenente al secondo corpus, attraverso una sommatoria dei valori dei loro pesi, ottenendo così la pagina la cui sommatoria presentava il valore più alto e quindi la candidata principale ad essere allineata.

```
trova_pag_corrisp(input: dataframe1, dataframe2):
    elem_imp = termini_importanti(dataframe1)
        // lista di termini con valore maggiore di 0.07 per ogni indice in dataframe1
    return allinea_pagine(dataframe2[elem_imp])
        // ritorna una lista di associazioni tra dataframe1 e dataframe2
        // in base all'indice che ha ottenuto il risultato maggiore in
        // base ad una sommatoria rispetto ai diversi indici
```

Questa operazione è stata applicata in entrambe le direzioni; una volta considerando le parole importanti per il primo dominio confrontate nel secondo e una volta in senso opposto. Si ottengono così due liste separate i cui elementi sono le corrispondenze tra le pagine di un sito web con l'altro, a seconda dei parametri in input. Queste due liste sono state successivamente confrontate per trovare le corrispondenze comuni ad entrambi i siti.

3.4 Complessità Computazionale

Nonostante la parte più critica sia lavorare con le foglie estratte dalle varie sorgenti, queste vengono poi filtrate ottenendo così un numero di termini comparabili con il numero di pagine nelle sorgenti stesse. La complessità computazione viene così ridotta, in quanto si passa da un insieme di termini dell'ordine di 10^3 ad un ordine di qualche centinaio di termini. L'operazione più onerosa risulta quindi essere quella sulla corrispondenza delle pagine (funzione trova_pag_corrisp) in cui la complessità è $O(n^3)$.

4 Valutazioni

Al fine della valutazione sono state utilizzate tre sorgenti che chiameremo S_1, S_2, S_3 rispettivamente: nba, rotoworld, espn. Le cardinalità di queste sono mostrate in Figura1.

Sorgente	Numero Pagine
S_1= nba.com	100
S_2= rotoworld.com	112
S_3= espn.com	105

Figure 1: Sorgenti e numerosità delle pagine

Definendo Rel come l'insieme delle pagine in comune tra due sorgenti, Ret come l'insieme delle corrispondenze restituite dal programma, $Rel \cap Ret$ come l'insieme delle corrispondenze corrette recuperate. Sono state calcolate le seguenti metriche Precision, Recall e F Measure definite come:

$$\text{Precision} = \frac{|Rel \cap Ret|}{|Ret|}$$

$$\text{Recall} = \frac{|Rel \cap Ret|}{|Rel|}$$

$$\text{F\_Measure} = \frac{2 * Recall * Precision}{Recall + Precision}$$

Nella Figura 2 sono indicati i valori Ground Truth e quelli restituiti dal programma.

Intersezione	Rilevanti (Rel)	Recuperate (Ret)	(Rel ∩ Ret)
$S_1 \cap S_2$	61	64	61
$S_2 \cap S_3$	51	53	50
$S_1 \cap S_3$	50	52	46

Figure 2: Valori Attesi e Restituiti

Per ogni coppia di sorgenti, attraverso l'utilizzo delle metriche sopra descritte, è stato calcolato il tasso di correttezza e quanto queste si discostavano dai valori predetti; in particolare si è notato come l'algoritmo ritornasse soluzioni migliori attraverso il confronto tra le sorgenti S_1 e S_2, ottenendo il 100% di Recall e recuperando soltanto tre corrispondenze errate. L'algoritmo invece ha restituito le prestazioni peggiori lavorando con le sorgenti S_2 e S_3, questo a causa dell'elevato rumore presente in tutte e due le sorgenti. Nonostante ciò i valori ricavati risultano essere comunque sufficientemente buoni. I risultati ottenuti sono consultabili nella Figura 3.

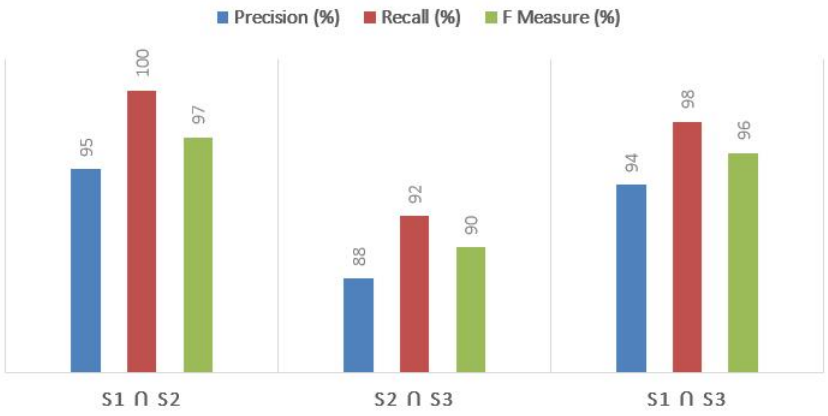

Figure 3: Risultati della Valutazione

5 Conclusioni e Sviluppi Futuri

L'obiettivo di questo lavoro è stata la realizzazione di un framework che, dati due insiemi di pagine da due sorgenti differenti, riuscisse ad allineare le pagine web riferite alla stessa entità.
La prima parte, quella di wrapping, si è focalizzata sull'estrazione mirata delle informazioni dalle pagine web. Queste, sono state poi pre-processate al fine di eliminare quei termini che non erano presenti in entrambi i siti e rendere computazionalmente meno onerosa l'elaborazione. Successivamente si è cercato di esaltare i termini importanti e dar meno valore a quelli che nonostante comuni risultassero inutili all'interno del contesto. L'ultima parte invece ha riguardato l'allineamento vero e proprio delle pagine attraverso l'utilizzo dei termini più importanti ed un matching incrociato.
Il progetto è stato reso il più possibile modulare in modo tale da facilitare un successivo sviluppo e ampliamento delle funzionalità del sistema.
Un possibile sviluppo futuro del lavoro riguarda l'implementazione di un algoritmo in grado di decidere automaticamente la giusta soglia da impostare per discriminare i termini più rilevanti, probabilmente mediante l'utilizzo del percentile o di una soglia ponderata in quanto la soluzione da noi proposta utilizza una soglia statica fissata ad un valore pari a 0.07. Questa è stata inferita empiricamente osservando che i termini con peso minore risultavano rumorosi e poco rilevanti al fine della determinazione della pagina corrispondente. Ciò nonostante è risultato evidente che se due giocatori non presenti in entrambi i siti condividevano nome o cognome, questi in alcuni casi venivano comunque accoppiati. In altri contesti ove questi attributi non risultano essere così discriminanti, l'algoritmo risulta essere ancora più efficiente.