Web-Scale Information Retrieval and Analysis

## Andrea Giorgi | Matricola 508800

## Filippo De Marco | Matricola 508304

# Introduzione

Il contesto applicativo di questo progetto risiede nella comparazione di dati prelevati da due sorgenti S1 e S2, in modo tale da poter rilevare coppie di pagine che fanno riferimento alla stessa entità. La soluzione identificata per questo problema è quello di utilizzare un crawler specializzato nel recupero delle pagine di interesse dalle due sorgenti, una volta recuperate le pagine di interesse queste verranno utilizzate dall’algoritmo di comparazione per restituire le coppie di pagine che trattano dello stesso elemento.

### Dominio applicativo

Il dominio preso in studio è quello del basketball, in questo dominio la nostra soluzione prevede di prelevare le informazioni da due sorgenti specializzate rispetto al dominio applicativo e di trovare coppie di pagine che trattano dello stesso giocatore. La prima sorgente S1 presa in studio è nba.com, mentre la seconda sorgente S2 è rotoworld.com; queste due sorgenti contengono al loro interno diverse tipologie di informazioni, sarà compito del crawler quello di andare a discriminare le pagine da recuperare, in modo tale da fornire all’algoritmo di comparazione solamente le pagine rilevanti rispetto al dominio.

### Analisi del problema

Il problema preso in studio è relativo alla comparazione delle pagine recuperate per individuare le pagine che trattano dello stesso giocatore, per poter far ciò sono state studiate diverse soluzioni applicative relative all’analisi del codice sorgente html. Questo studio si è focalizzato sull’utilizzo del codice sorgente di ciascuna pagina, andando ad utilizzarne i tag notevoli rispetto al dominio recuperandoli tramite l’uso di XPath.

Una volta identificata una modalità di soluzione è stato necessario individuare le tecnologie più adeguate per risolvere la task data. Il linguaggio di programmazione adoperato è stato Python, il quale è perfettamente adattabile a problemi nei quali è necessario modellare dei documenti generici. Le varie librerie che sono state inserite all’interno dei vari file ne permettono il corretto funzionamento, nello specifico permettono di lavorare utilizzando dei file in formato html e di riuscire ad analizzarli a dovere.

# Soluzione

La soluzione del problema è stata progettata cercando di dividere in tre macroaree le varie operazioni da svolgere:

1. Allineamento delle pagine
2. Rappresentazione delle pagine, nel nostro caso tramite il modello TF-IDF
3. Valutazione dei risultati

### Allineamento delle pagine

L’allineamento delle due sorgenti informative, che chiameremo S1 ed S2 per semplicità, è l’operazione principale della soluzione individuata. Questo algoritmo ci permette di esaminare le pagine web delle due sorgenti e di individuare quelle che presentano delle label in comune utilizzando delle funzioni apposite.

La prima funzione ausiliaria implementata è la funzione relativa al calcolo dell’intersezione, questa funzione scorre le liste di label delle due sorgenti cercando delle corrispondenze, se individua una corrispondenza andrà ad incrementare un valore numerico che permette di tenere traccia delle intersezioni tra le due pagine.

Lo pseudocodice della funzione è il seguente:

DEF intersect(S1Labels, S2Labels)

intersection = 0

FOR label1 in S1Labels

FOR label2 in S2Labels

IF label1 is equal to label2

THEN intersection + 1

La seconda funzione ausiliaria implementata è adibita al filtraggio degli elementi in base al loro valore associato di TF-IDF. Grazie a questa operazione siamo in grado di eliminare dalle label da valutare quelle con basso valore di TF-IDF, questo perché al diminuire del valore diminuisce anche l’importanza del termine all’interno del corpus in analisi.

Lo pseudocodice della funzione è il seguente:

DEF filter (Source, minValue)

Values = []

FOR key in S.items()

IF S[key] > minValue

THEN values.add(key)

La funzione principale è allignSources, ed è la funzione che implementa la parte di scansione delle due sorgenti informative. Vengono scorse entrambe le sorgenti andando a cercare per ciascun elemento di S1 una possibile corrispondenza con un elemento di S2, questo avviene tramite l’utilizzo delle due funzioni ausiliarie. Viene impostato un valore minimo di intersezione pari ad 1 per scongiurare la presenza di coppie fasulle.

La classe che implementa queste tre funzioni è PageAllignament e ritorna una lista di coppie di pagine web che dovrebbero essere relative alla stessa entità.

Lo pseudocodice della funzione è il seguente:

DEF allignSources(S1, S2)

Pairs = []

FOR i in [0, len(S1)]

elementS1 <- S1[i]

matches = []

filteredS1 <- filter(elementS1)

FOR j in range [0, len(S2)]

elementS2 <- S2[j]

filteredS2 <- filter(elementS2)

matches.append(intersect(filteredS1, filteredS2))

maxMatch = max(matches)

IF maxMatch > MIN\_INTERSECTIONS #Il cui valore minimo è 1

Pairs.add((i, matches[maxMatch])

### Modellazione delle pagine tramite il modello TF-IDF

La sigla TF-IDF significa “Term Frequency – Inverse Document Frequency” e rappresenta una formula matematica per calcolare l’importanza di una parola inserita in un documento in rapporto ad altri documenti simili. Viene utilizzato nei contesti applicativi di Information Retrieval e proprio per questo è perfettamente adattabile a questo dominio applicativo.

Il valore TF-IDF aumenta in modo proporzionale al numero di occorrenze di un dato termine, valore che viene compensato dalla frequenza del termine nel corpus di documenti.

Per poter implementare questa modellazione è stata utilizzata la libreria sklearn, la quale fornisce strumenti specializzati nel campo dell’intelligenza artificiale e del machine learning. Nello specifico è stato utilizzato un TF-IDF Vectorizer, il quale a partire da dei documenti grezzi genera una matrice di feature TF-IDF.

La modellazione segue i seguenti passi fondamentali:

* Creazione di un dizionario vectorizedHTML a partire dai vettori generati dal vectorizer
* Controllo delle chiavi del dizionario, per assicurarci di non aver inserito delle chiavi al posto dei valori associati
* Ordinamento del dizionario per rispettare l’ordine delle pagine html date in input, questo perché le pagine che verranno usate per l’esecuzione sono ordinate alfabeticamente.

### Valutazione dell’algoritmo

La procedura di valutazione dell’algoritmo viene eseguita da una classe apposita, la quale prende come dati il numero totale di coppie generate dall’algoritmo e va a controllarne la veridicità.

A partire da questi dati verranno calcolate le metriche di valutazione più comuni, ossia:

* Precision, è la probabilità che un documento recuperato sia attinente al dominio;
* Recall, è la probabilità che un documento attinente venga recuperato;
* F1 Measure, è la misura armonica della Precision e della Recall.

### Analisi della complessità

La complessità dell’algoritmo implementato è principalmente determinata dalla funzione di lettura delle due sorgenti informative, dunque nel caso peggiore abbiamo una complessità temporale e spaziale pari a O( dove n è la dimensione di S1 ed m la dimensione di S2.

Tutte le altre operazioni verranno effettuate effettivamente sia n che m volte, se supponiamo S1 e S2 di egual dimensione possiamo dedurre che l’operazione di vettorizzazione, di allineamento e di filtraggio avranno sempre complessità pari a O(

# 3. Attività sperimentale

L’attività sperimentale di valutazione della soluzione implementata si è basata sulla valutazione dell’algoritmo utilizzando una terza sorgente non nota, che chiameremo S3. Questa sorgente è il sito sportivo <https://basketball.realgm.com/>, specializzato nel fornire informazioni relative al mondo del basketball.

Come prima operazione effettuata è quella del recupero delle pagine relative ai vari giocatori di basket, raccogliendo cento campioni da S3. Questa operazione è stata effettuata anche per S1 ed S2 per ottenere un dataset adeguato alla fase di valutazione sperimentale.

L’algoritmo è stato avviato due volte, analizzando prima S1 e S3 e poi S2 e S3, andando a valutarne le prestazioni in entrambi gli scenari.

### NBA e REALGM

In questo scenario l’algoritmo è riuscito a determinare 45 coppie di pagine che trattano dello stesso giocatore di basketball, ottenendo rispetto alle metriche di valutazione i seguenti valori:

* Precision = 0.91
* Recall = 0.9
* F1 = 0.9

### Rotoworld e REALGM

In questo scenario l’algoritmo è riuscito a determinare 35 coppie di pagine che trattano dello stesso giocatore di basketball, ottenendo rispetto alle metriche di valutazione i seguenti valori:

* Precision = 0.8
* Recall = 0.7
* F1 = 0.74

### NBA e Rotoworld

In questo scenario l’algoritmo è riuscito a determinare 5 coppie di pagine che trattano dello stesso giocatore di basketball, ottenendo rispetto alle metriche di valutazione i seguenti valori:

* Precision = 0.62
* Recall = 0.58
* F1 = 0.59

# Sviluppi futuri

L’algoritmo implementato presenta diversi margini di miglioramento, il primo fra tutti è quello di affiancargli un sistema di crawling specifico che permetta un recupero automatico delle pagine appartenenti al dominio. Utilizzando questa tecnologia di crawling, e raffinando il processo di allineamento delle pagine, potrebbe essere possibile sviluppare un applicativo specifico per il mondo del sport.

Un esempio concreto di sviluppo futuro è quello della realizzazione a partire da questo progetto di un applicativo relativo al fantacalcio, gioco di fantasia che prevede di realizzare una propria squadra partecipando ad un’asta con tutti i giocatori presenti nel campionato di calcio d’interesse. L’utilizzo di questo applicativo permetterebbe di recuperare in tempo reale le informazioni di un dato giocatore da diverse fonti, fornendo così un servizio di raccomandazione all’utente finale.