Università degli Studi di Padova

Attività formativa: elaborato

Da una seed list al crawling e al grafo del web italiano

Relatore: Prof. Luca Pretto
Correlatore: Prof. Enoch Peserico

A.A. 2009/2010

Laureando: Marco Compagnin - 582188 IF

Corso di laurea in Ingegneria Informatica

28 settembre 2010

Sommario

I sistemi di Information Retrieval tradizionali operano su collezioni di documenti di qualità omogenea, e hanno l'unico scopo di restituire i documenti più rilevanti alle interrogazioni formulate dagli utenti. L'applicazione delle tecniche di Information Retrieval tradizionale al web si è rivelata poco efficace, sostanzialmente a causa dell'eterogeneità della qualità delle pagine web; di qui la necessità di ideare algoritmi in grado di selezionare le pagine web anche in base alla loro qualità. Tra questi algoritmi, un posto di rilievo è stato assunto da quelli di link analysis, che cercano di inferire la qualità delle pagine web dalla struttura topologica del grafo associato al web.
Il lavoro descritto in questa relazione è stato svolto all'interno di un progetto che ha lo scopo di valutare l'effettiva efficacia degli algoritmi di link analysis nell'individuare le pagine web di maggiore qualità. Il nostro lavoro è consistito nel contribuire al crawling esaustivo del web italiano per costruirne l'intero grafo; in particolare, ci siamo occupati della scelta di un insieme ottimale di documenti da cui esso ha avuto inizio (seed list) e della gestione del crawling.
Per quanto riguarda la seed list è stato costruito un insieme di risorse, composto di servizi gratuiti di directory web popolari, nelle quali sono presenti un gran numero di link a risorse italiane. Per completare quest'insieme è stata valutata la possibilità di utilizzare alcuni motori di ricerca per ottenere altri link a risorse del dominio italiano.
Per quanto riguarda il crawling è stata scelta una configurazione del software utilizzato che realizzasse politiche di comportamento adeguate per un'analisi efficace e robusta delle risorse web e raggiungere inoltre l'obiettivo prefissato.
Durante il procedimento di crawling una procedura automatizzata permette di scrivere su file la parte di grafo relativo alle risorse analizzate.

Indice

Capitolo 1

Introduzione al progetto

Il lavoro sviluppato in questa relazione si inserisce all'interno di un progetto dove si vuole valutare la qualità delle pagine web determinata dagli algoritmi di lynk analysis. Questo giudizio sarà confrontato con il parere di risorse umane, cui si sottopone un insieme di pagine web rappresentativo del web italiano per ottenere attraverso votazione una valutazione di qualità delle stesse.
Per raggiungere le distinte valutazioni di qualità delle pagine web, il progetto si suddivide in due parti sviluppate ognuna da un team diverso.
Uno di essi ha il compito di campionare le risorse del web rese dall'esecuzione di interrogazioni costruite in precedenza e sottoposte ai motori di ricerca. Inoltre gestisce un'applicazione in grado di scaricare alcune pagine web italiane e un'applicazione che permette alle risorse umane di valutare le pagine web scaricate.
Il secondo team si pone l'obiettivo di eseguire il crawling di tutto il web italiano al fine di ottenere il grafo associato.
Questo scritto è riferito a quest'ultima parte del progetto e nello specifico mette in evidenza la metodologia utilizzata per realizzare il crawling del web italiano inteso come dominio .it.
La relazione è in particolar modo inerente alla progettazione di un insieme di indirizzi URL da cui far procedere un'istanza di crawling eseguendo una corretta scansione delle pagine e applicando le politiche necessarie al caso, al fine di raggiungere tutte le risorse del web italiano. Durante la scansione delle risorse viene eseguita una procedura per la scrittura di file dove verrà creata una rappresentazione del grafo del web italiano.
Il progetto si concentra sul web italiano principalmente per i seguenti tre motivi:

- il web italiano è isolato sia perché esso non risulta largamente puntato da risorse esterne al dominio, sia perché le risorse che appartengono al

web italiano non puntano in modo elevato a risorse non italiane. Inoltre è ragionevole pensare che la struttura del web italiano sia simile alla struttura di tutto il web data l'auto-similarità del web [13];

- le dimensioni stimate consentono di effettuare un crawling esaustivo;
- ragioni di lingua rendono più agevole ai valutatori fornire giudizi sulla qualità delle pagine.

Per ottenere un'immagine aggiornata e recente del web italiano è necessario utilizzare il crawling grazie al quale è possibile ottenere una rete topologica, rappresentata dal grafo del web, simile a quella reale.
La valutazione qualitativa delle risorse web è giustificata dal fatto che anche persone non autorevoli possono pubblicare materiale di qualità eterogenea nella rete mondiale. Il progetto di ricerca non trova riscontro in nessun'altra trattazione esistente.

Capitolo 2

Teoria preliminare per il crawling

Verranno qui presentate l'insieme di conoscenze teoriche sul web in generale, sul crawling e sul software utilizzato, necessarie a comprendere lo sviluppo del progetto.

2.1 Web: collezione di risorse

La rete mondiale può essere considerata come un insieme molto grande di risorse eterogenee che è sottoposto ad un continuo e rapido sviluppo.
Anche il web italiano segue questo sviluppo incrementando i domini italiani registrati come documentato in [5].
Con l'avvio della rete mondiale si è passati da un sistema contenente documenti di qualità elevata ad un insieme di risorse di numero elevato che possono essere realizzate facilmente da qualsiasi persona anche non competente e la cui qualità non è assicurata.
Per poter raggiungere un qualsiasi documento nel web si deve conosce l'indirizzo URL al quale risiede.

2.1.1 URI e URL

Nel corso della trattazione si useranno spesso questi due termini per poter indicare su quali oggetti agisce il crawler.
Il termine URL è l'acronimo di Uniform Resource Locator e indica una sequenza di caratteri che identifica in modo unico all'interno del web l'indirizzo al quale è possibile reperire una data risorsa. Sarà attraverso questi indirizzi che il crawler potrà spostarsi e raggiungere tutto il web italiano.

Un URL ha una struttura standard che permette non solo di poter effettuare delle analisi testuali e selezionare tra gli URL raggiunti da una scansione tutti quelli di un dato dominio, ma è possibile anche selezionarli in base al protocollo utilizzato.
Il termine URI equivale a Uniform Resource Identifier e identifica univocamente un contenuto su Internet, sia esso un file di testo, un'immagine, un video, un programma o quant'altro. Un URI generalmente descrive il meccanismo per poter accedere ad una risorsa web e specifica in quale elaboratore e in quale percorso essa è contenuta.
Ad esempio $''http://''$ identifica il protocollo da utilizzare per accedere alla risorsa, $''www.gazzetta.it''$ identifica il computer dove la risorsa è salvata $''images/logos/logo_normale.gif''$ indica il percorso e il nome della risorsa all'interno dell'elaboratore.

2.1.2 Pagine statiche e dinamiche

Come citato in [14] le pagine che costruiscono il web si possono suddividere in due categorie distinte tra loro in base al tipo di servizio che forniscono all'utente che le consulta.
Vi sono le pagine statiche che sono dei file che risiedono in uno spazio web e vengono scaricate semplicemente quando un utente le richiede. Esse non cambiano durante la richiesta di un utente e sono rappresentate da un singolo documento.
Le pagine dinamiche invece hanno un contenuto variabile in base alla richiesta fatta dall'utente ed elaborata poi dal server. Il loro URL contiene nella maggior parte dei casi una forma standard per descrivere l'interrogazione a cui si dà risposta.
In base alla tipologia di servizio fornito da parte di una risorsa l'URL che l'identifica risulta essere il medesimo per tutte le risorse web di tipo statico, mentre cambia per ogni elaborazione distinta richiesta su una stessa risorsa dinamica.
Il crawler deve quindi mantenere un comportamento che consideri o meno di questa diversità di identificazione degli indirizzi dei due tipi di documenti web per tutte le risorse scansionate.

2.1.3 Struttura del web

Anche se il web è una risorsa illimitata, eterogenea e disordinata può essere comunque delineata una sua struttura anche se essa non è rigorosa.

Tutte le pagine web di qualsiasi tipo esse siano, possono essere ricondotte ad una delle quattro tipologie possibili individuate nell'articolo [7] .

- Central/core pages: pagine che possono essere navigate facilmente dall'utente.
- In pages: l'insieme delle pagine che contengono link verso pagine core centrali, quindi si potrà raggiungere da una risorsa di quest'insieme una pagina centrale.
- Out pages: insieme delle pagine web che sono linkate dalle pagine core e quindi da esse raggiungibili.
- Disconnected pages: pagine che non sono raggiungibili attraverso link in uscita da nessuna delle tre tipologie di pagine precedenti.

I link tra le diverse risorse del web possono essere così descritti:

- Tendrils: link di tipo terminale e quindi che non collegano altre pagine o risorse, ad esempio quando una pagina punta a una pagina web senza altri link o una risorsa di sola consultazione.
- Tubes: link che rappresentano una connessione ad altre pagine e non sono quindi terminali.

Le categorie di pagine e link vengono rapidamente riassunte dall'immagine 2.1 relativa ad uno studio classico sul grafo del web. Il web risulta essere un insieme di documenti in continua evoluzione, con modifiche, cancellazioni e nuovi inserimenti attraverso i quali struttura e grandezza certe diventano impossibili da determinare.
Non risulta inoltre possibile pensare di poter conoscere la totalità dei siti web, non solo per quanto riguarda la loro variabilità temporale ma anche perché una parte definita deep web risulta essere irraggiungibile dal crawling.

2.1.4 Ipotesi di grafo

Dopo aver studiato come potrebbe essere delineata la struttura del web si ipotizza come si possa mappare in un grafo l'insieme dei documenti che si andranno ad analizzare ed i loro collegamenti.
La produzione finale del progetto è il grafo del web italiano, esso sarà costituito da un insieme di nodi interconnessi tra di loro dove il significato attribuito a ciascun elemento viene ora descritto:

- nodo: ogni nodo del grafo viene associato ad una risorsa web che potrà o meno avere dei link in uscita e dei link in entrata;

- arco: ogni arco che interconnette una coppia di nodi tra loro viene associato a un link tra due risorse nel web.

Il grafo dovrà esprimere il fatto che una risorsa possiede dei link in uscita verso un'altra risorsa, in modo tale che si possa distinguere attraverso una relazione di tipo padre-figlio quale risorsa è puntata e quale punta ad un'altra. Risulta necessario che il grafo sia un grafo orientato e che quindi tutti gli archi presenti possiedano un verso.
Non interessa per questo progetto associare per ora oltre a un nome univoco altri punteggi o altri campi ad ogni nodo e quindi il grafo che si vuole creare non è un grafo arricchito.
Come nella struttura del web esposta nel sottoparagrafo si è certi che il grafo del web italiano prodotto possa presentare dei nodi senza archi in uscita o dei nodi privi completamente di archi quindi isolati. Il grafo rispecchiando le proprietà del web risulta essere non fortemente connesso poiché non tutti i nodi sono connessi l'uno all'altro e non esiste un percorso che li possa collegare tutti.

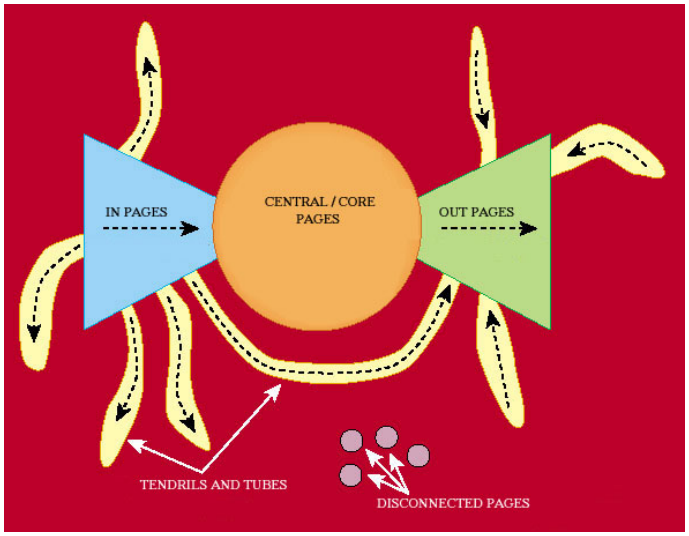

Figura 2.1: Struttura del web

2.2 Crawling del web

Il crawler è un software per scaricare delle pagine web che può essere utilizzato per l'analisi delle stesse.

Esso permette di effettuare il download delle pagine web e possibilmente salvarle in memoria per analizzarle utilizzando degli strumenti adatti in base alle risorse presenti nella pagina e al tipo di documento recuperato. Successivamente esso ricerca ed estrae tutti i link presenti all'interno della risorsa analizzata.
Proprio questi link risultano essere gli indirizzi delle risorse web che vengono utilizzate dal crawler per espandere la sua ricerca su altri documenti; in una successiva fase infatti saranno analizzate proprio le risorse di cui sono stati reperiti gli indirizzi web per cercare altri link e far avanzare la ricerca.
Il crawler quindi utilizza gli URL web contenuti nei link delle risorse che vengono scansionate per continuare la sua analisi.

2.2.1 Seed list

Il software di crawling deve avere però un input iniziale suggerito da un operatore che segue il lavoro di crawling.
Questo input si configura in una lista di indirizzi URL di risorse web da cui poter iniziare l'analisi ed effettuare poi l'istanza di crawling. La scelta della seed list risulta essere molto importante per i seguenti motivi:

- dalla lista iniziale dipende il risultato di tutto il crawling come numero totale di pagine che si possono ottenere dalla ricerca (ad esempio: pagine senza link)
- la lista di pagine iniziali configura la tipologia di documenti web che verranno scansionati dal crawler, in particolare ad esempio ci saranno collezioni iniziali studiate per una certa portata del crawling (ad esempio: il web italiano);

Definita la struttura del web precedentemente si può comprendere come ci siano delle risorse disconnesse da tutte le altre che rappresentano una particolare difficoltà se si vuole conoscere la totalità delle pagine web; queste risorse saranno infatti raggiungibili solamente da un indirizzo URL diretto e non potranno essere raggiunte da nessun altro link contenuto nelle altre risorse scansionate.
La scelta quindi della collezione iniziale di risorse del web da cui far partire il progetto di crawling è stata progettata appositamente per il progetto e sarà oggetto di approfondimenti nel corso di questa relazione.

2.2.2 Motori di ricerca

Un motore di ricerca è un sistema automatico che analizza le risorse web e le indicizza e categorizza in modo da poter rispondere a una richiesta qualsiasi

dei navigatori web.
La maggior parte dei motori di ricerca che operano sul web sono gestiti da enti privati e per svolgere il loro compito utilizzano algoritmi e strumenti di memorizzazione mantenuti segreti.
Il lavoro che svolgono si caratterizza in una prima fase di analisi del web attraverso la tecnica di crawling, seguita poi da una memorizzazione dei contenuti analizzati su database e directory, per poi passare alle operazioni di indicizzazione.
I motori di ricerca mantengono la fotografia più accurata e più aggiornata del web, e oltre al crawling periodico reperiscono le risorse web anche attraverso la segnalazione spontanea da utenti interessati a veder indicizzati i loro contenuti web.
La prima fase di analisi è presente anche in questo progetto ma si contraddistingue per la ricerca focused solamente sul web italiano.
Si può comunque cercare di comprendere come siano costituiti gli elementi della seed list dei motori di ricerca e se i motori di ricerca stessi possono essere utili al progetto.

2.2.3 Tipi di crawling

Ci sono diverse tipologie di processi di crawling in base all'estensione della porzione del web che si vuole analizzare e alla modalità con cui si effettua il procedimento. Le tipologie si potrebbero classificare in:

- crawling tradizionale: viene visitato l'intero web cioè l'insieme di tutte le risorse raggiungibili dalla seed list indistintamente dall'argomento trattato o da altre caratteristiche peculiari delle risorse scansionate;
- crawling periodico: viene ispezionata una porzione del web e periodicamente si mantengono aggiornati i dati su di essa effettuando nuovamente il crawling;
- crawling incrementale: vengono selezionate parti del web distinte e si effettua il crawling di queste in modo successivo;
- focused crawling: una tipologia di crawling che visita una certa porzione di web in base a un certo argomento tralasciando le risorse esterne a esso.

Tra le diverse possibilità di crawling il progetto impone l'utilizzo di una tipologia focused che si otterrà in base alla configurazione del software di crawling utilizzato e alla seed list progettata.

2.2.4 Politiche di crawling

Per un buon progetto di crawling bisogna essere in grado di poter impostare il software utilizzato in rispetto a quattro politiche di comportamento.
Queste quattro politiche riguardano non solo il software, ma anche le risorse interne utilizzate e le risorse dei server web contattati.

- Politica di selezione: deve essere possibile impostare nel crawler un comportamento tale da poter indirizzare la portata della scansione verso la parte di web obiettivo del progetto.

- Politica di rivisitazione: deve essere possibile determinare come il software si comporti in base alla presenza di una molteplicità di medesimi indirizzi all'interno della collezione di risorse web da analizzare. Si può pensare di rivisitarli, di non rivisitarli, di rivisitarli periodicamente per verificarne i cambiamenti.

- Politica di cortesia: usare un crawler per scansionare un insieme di risorse su un server web può compromettere le prestazioni dell'elaboratore visitato. Uno dei casi più frequenti è la saturazione della banda del server web analizzato. Per questo motivo risulta necessaria la possibilità di impostare dei vincoli per limitare la visita troppo veloce di pagine di uno stesso server web.
  Nella rete inoltre sono presenti delle limitazioni sulle pagine consultabili da un utente, a questo proposito dovrebbe essere possibile impostare nel crawler la possibilità di seguire o meno queste limitazioni imposte dai server web.

- Politica di parallelismo: il software di crawling dovrebbe poter essere utilizzato in modo efficiente rispetto alle risorse messe a disposizione nell'elaboratore che lo ospita ad esempio attraverso il parallelismo, diminuendo lo spreco di banda e aumentando le prestazioni nell'analisi dei documenti.
  Deve essere inoltre possibile determinare quando una risorsa non può essere analizzata creando degli errori tali che il crawling non può procedere oltre ed è quindi necessario scartare la risorsa che li provoca.

Queste politiche condizionano anche la buona riuscita del crawling e verranno impostante nel software utilizzato.

2.2.5 Politica di selezione e visita delle risorse

La modalità con la quale si compie l'analisi delle risorse web definita nella politica di selezione può essere effettuata in due modi diversi:

- breadth-first: questo tipo di analisi si pone l'obiettivo di scansionare tutti i link uscenti da una certa risorsa web e proseguire ricorsivamente la scansione di ognuno di questi fino a un solo livello di profondità. In questo modo si predilige la ricerca in ampiezza dei link. La figura 2.2 sotto riportata visualizza come avviene la visita;

- depth-first: al contrario del metodo precedente questa visita è basata sulla profondità, in particolare data una risorsa vengono analizzati tutti i link in uscita e sequenzialmente per ognuno di questi si analizzano i successivi link uscenti dal primo all'ultimo ricorsivamente fino a che non ci sono più link in uscita. La figura 2.3 sotto riportata visualizza il concetto di visita in profondità.

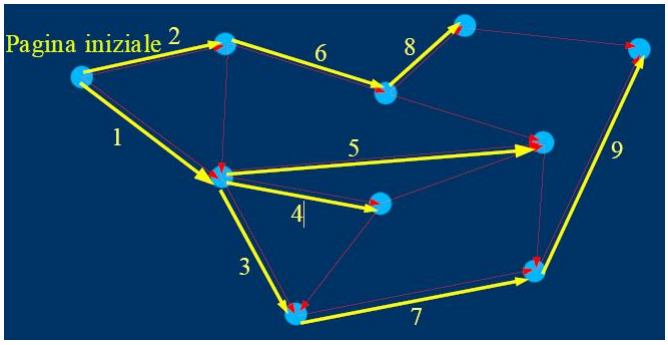

Figura 2.2: Visita breadth-first delle risorse

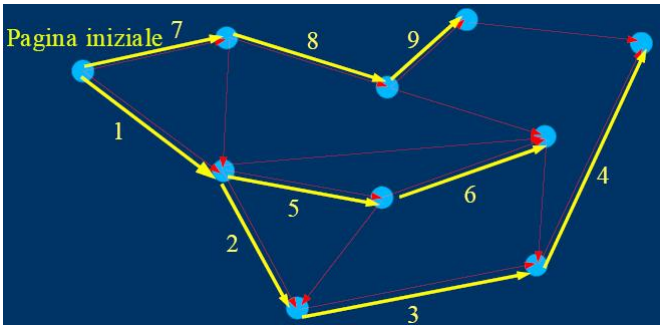

Figura 2.3: Visita depth-first delle risorse

2.2.6 Politica di rivisitazione e grafo

Per la costruzione del grafo del web si vuole associare univocamente un nodo a un URI analizzato dal crawler.
La politica di rivisitazione può influenzare la costruzione del grafo perché abilitando una procedura automatizzata per l'associazione tra risorsa scansionata ed entità topologica del grafo si potrebbero creare delle duplicazioni nel caso una risorsa possa essere visitata più di un'unica volta.
Si deve quindi decidere se impostare una rivisitazione univoca delle risorse da parte del crawler o creare delle procedure automatizzate per la scrittura del grafo che riconoscano se la parte del web in analisi è già stata scritta o meno.
Entrambe le modalità garantiscono l'univocità dell'associazione tra risorsa web ed elemento del grafo, resta da decidere quale sia tra le due quella più efficiente e più adatta al progetto.

2.2.7 Politica di cortesia e file robots.txt

Per impostare la politica di cortesia del crawler bisogna essere a conoscenza dell'utilizzo dei file *"robots.txt"* presenti negli spazi dei siti web.
Al suo interno sono scritte delle direttive per permettere l'analisi di qualsiasi crawler su certe pagine o percorsi dello stesso sito web, oppure per vietare l'analisi su certi percorsi o certe pagine ritenute private o segrete e che il gestore del server web non vuole che siano analizzate.
In questo file inoltre è possibile inserire delle direttive per vietare completamente l'analisi del sito web a diversi tipi di crawler attraverso l'header con cui questi software si identificano.
Anche l'aspetto temporale può essere inserito all'interno di questo file. Attraverso direttive temporali nella versione estesa degli standard di *"robots.txt"* risulta possibile definire quale sia l'intervallo minimo per la visita delle pagine di uno stesso sito web e l'orario rispetto al GMT nel quale è possibile effettuare la visita delle risorse stesse.

```
1 User-agent: *
2 Disallow: /private_directory/   //Blocca la directory /private_directory/
3 Request-rate: 1/5               //Visita al massimo una pagina ogni 5 ↩
      secondi
4 Visit-time: 0600-0845           //Visita soltanto tra le 6:00 AM e le 8:45↩
      AM (GMT)
```

Listing 2.1: Esempio direttive robots.txt esteso

Buona parte della politica di cortesia si basa sull'impostazione o meno dell'aderenza del crawler proprio alle regole impostate dal file *"robots.txt"*.

2.2.8 Politica di parallelismo e problemi di scansione

Come descritto nella politica di parallelismo il software potrebbe non essere in grado di effettuare la scansione di una determinata risorsa web.
I motivi principali sono due:

- time-out: la connessione a un server web risulta impiegare troppo tempo senza dare una risposta, e quindi probabilmente per causa del traffico di rete il pacchetto di risposta o di richiesta viene perso e la connessione risulta in time-out;
- spider-traps: probabilmente per proteggersi dalle continue scansioni da parte di diversi tipi di crawling è possibile che un sito web contenga pagine che non possono essere scansionate dal crawler o che scansionate siano progettate in modo tale da comportare un notevole spreco di tempo nella loro analisi.

Il software deve essere configurato anche in modo tale da ovviare se si presentassero dei problemi di questa natura garantendo rapidità ed efficacia per non impiegare troppo tempo nella scansione.

2.3 Software Heritrix

Il software per effettuare il crawling utilizzato è Heritrix, un progetto open source in continua evoluzione composto da un insieme di classi java chiamate moduli che permette di effettuare crawling su larga scala. La versione del software utilizzata è la 1.14.4 datata maggio 2010.

2.3.1 Funzionamento iterativo

Il software Heritrix per il suo scopo utilizza alcuni moduli importati in fase di creazione dell'istanza di crawling suddividendone il funzionamento in una sequenza di passi iterativi a cui sottoporre ogni URI analizzato. In casi particolari la sequenza di funzionamento può interrompersi o saltare un insieme di passi successivi, comunque sarà sempre effettuato lo step iniziale e finale.
I passi sono suddivisi come in figura 2.4.
Ogni passo elabora in un modo diverso gli URI e in particolare i compiti svolti sono principalmente i seguenti:

- **Pre-fetch:** insieme di moduli che indicano se gli URI possono o meno essere istanziati per il crawling, controllando l'indirizzo DNS e il file $''robots.txt''$ dello spazio web interessato.
- **Fetch:** insieme di moduli che si occupano di effettuare il trasferimento e il reperimento fisico delle risorse nel web per le future scansioni.
- **Extractor:** moduli per l'estrazione di tutti i link dai documenti reperiti in base alle configurazioni e alla tipologia degli stessi.
- **Writer:** questi moduli vengono utilizzati nella configurazione di default per poter scrivere i dati e le risorse analizzate in archivi compressi in formato ARC.
- **Post-processing:** questa fase è un processo obbligato per tutte le risorse ritenute valide, quindi analizzate, e quelle non analizzate perché non valide in base a vincoli espressi dall'utente; i moduli in questa tipologia si occupano di effettuare i necessari aggiornamenti per successive scansioni, di controllare se i link estratti sono nella portata del lavoro di crawling definito e di effettuare il lavoro di scheduling per le scansioni future sugli URI.

Il funzionamento iterativo per ogni lavoro di crawling da parte del software risulta invece essere indicato in modo esplicativo dalla seguente immagine 2.5.

Viene indicato come partendo dalla seed list iniziale di indirizzi di risorse web tutti gli URL sono passati direttamente alla frontiera, un modulo che si occupa di controllare che ogni risorsa analizzata sia effettivamente una buona candidata in base a filtri e altre impostazioni delineate dall'operatore al momento di avvio del job di crawling e che sia nella portata definita del progetto.
Ogni URI che passa questa selezione diventa quindi un candidato che può essere elaborato direttamente da un toe-thread specifico e sottoposto a tutta la catena in cui sono suddivisi i moduli visti in precedenza estraendone in particolare nuovi URL che vengono sostanzialmente utilizzati inviandoli al modulo frontiera come nuovi candidati. Il modulo frontiera fa parte del passo iniziale che tutte le risorse devono oltrepassare per diventare dei candidati ed essere analizzati.
Il funzionamento successivo segue quello iniziale per ogni URL reperito.

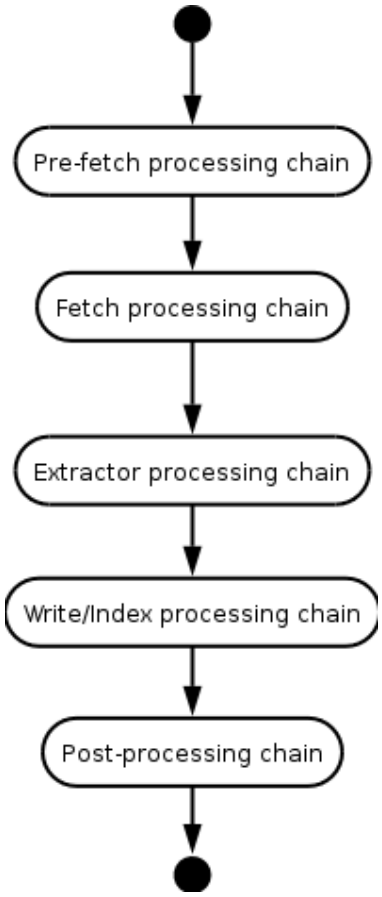

Figura 2.4: Passi per elaborare il crawling, suddivisione dei moduli per funzionalità

2.3.2 Aderenza al progetto

Il software Heritrix presenta delle caratteristiche che lo rendono molto utile e adatto per il progetto realizzato. Per prima cosa il software permette di scegliere una configurazione per ogni politica di crawling espressa precedentemente:

- riguardo la politica di selezione il modulo frontiera permette di determinare le condizioni per effettuare la scansione del solo web italiano basate sull'analisi testuale degli indirizzi scansionati;

- riguardo la politica di rivisitazione risulta possibile impostare solamente la rivisitazione o la non rivisitazione delle risorse già scansionate;

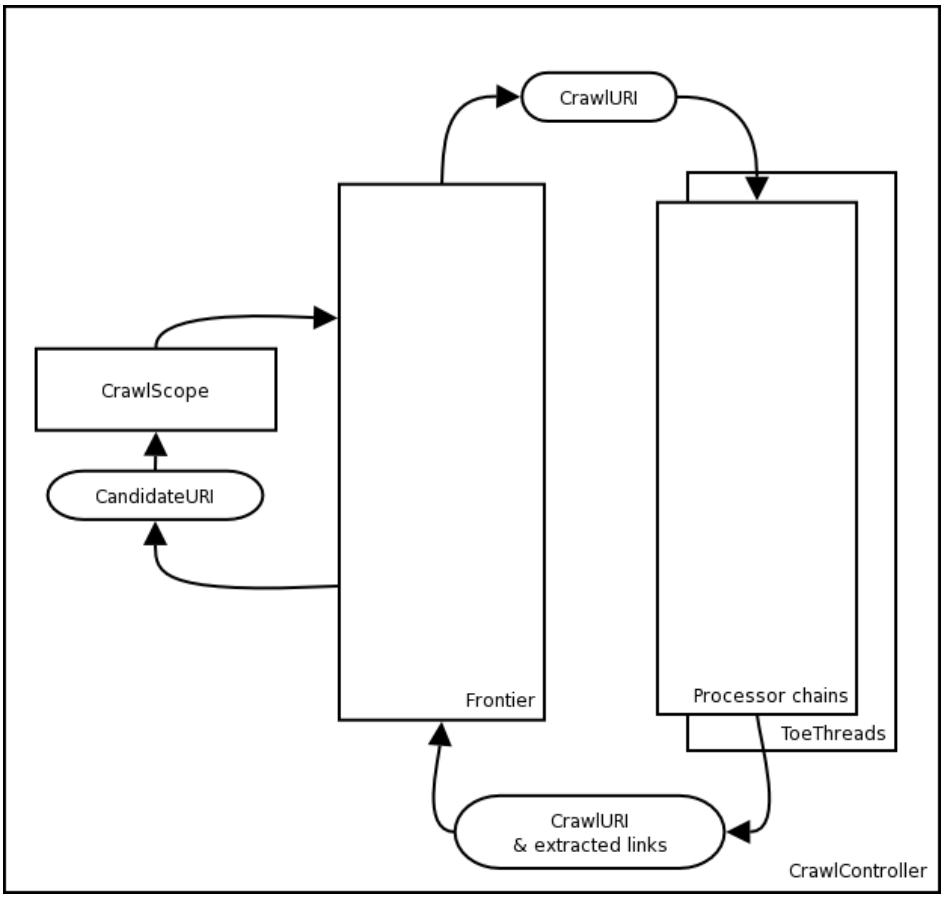

Figura 2.5: Funzionamento generale: passaggio degli URI da analizzare

- rispetto la politica di cortesia le impostazioni di configurazione permettono di attenersi o meno ai file *"robots.txt"* e di impostare dei timer per spaziare temporalmente l'accesso a risorse di uno stesso sito web;
- considerando la politica di parallelismo il funzionamento del crawler è di tipo multi-threaded e inoltre è possibile impostare dei timer per evitare che una connessione a uno stesso sito web duri per un tempo indefinito e configurare un numero massimo di tentativi per provare ad analizzare una risorsa web.

Inoltre la suddivisione in moduli del software permette di aggiungere funzionalità specifiche come la scrittura del grafo del web peculiare del nostro progetto.

Capitolo 3

Seed list, impostazioni utilizzate e grafo

In questa sezione si documentano in dettaglio la fase di costruzione dell'insieme degli elementi di partenza, le impostazioni utilizzate e le implementazioni di codice create per poter procedere al crawling del web italiano e alla produzione del corrispondente grafo.

3.1 Creazione della seed list

Come detto precedentemente l'insieme di URL inseriti all'avvio del crawling risulta essere molto importante e condiziona la buona riuscita di tutto il procedimento di crawling del web.
Attraverso ricerca sul web è risultato difficile trovare materiale specifico al caso di crawling del web italiano trattato ed è quindi stato necessario riadattare le informazioni reperite alla portata del progetto stesso; per i progetti di crawling di cui sono stati trovati dei documenti di presentazione la lista iniziale di URL risulta sempre essere esposta parzialmente oppure non esposta affatto.
Viene ora presentata l'analisi di documenti e le considerazioni effettuate per la creazione della seed list del progetto ponendo l'obiettivo di partire dalla seed list per arrivare a conoscere tutto il web italiano.

3.1.1 Lista di URL: politica del TLD italiano

Il dominio web italiano .it in quanto top level domain (TLD) risulta essere intestato alla Registration Authority Italiana presso il CNR di Pisa il cui sito web [5] propone statistiche relative al web italiano e in particolare viene

segnalato pubblicamente il numero di domini registrati al variare degli ultimi 10 anni e l'elenco di registers e manteiners .it.
Non risulta essere presente nel sito citato una lista completa di tutti i possibili indirizzi web dei domini italiani registrati. Se fosse possibile ottenere questa lista e utilizzarla come seed list si avrebbe la certezza di poter ottenere il grafo completo di tutto il web italiano.
Non riuscendo a ottenere queste informazioni perché riservate si dovrà trovare un insieme di siti web dai quali cercare di ottenere tutto il web italiano.

3.1.2 Scraping dei risultati da motori di ricerca

Scartata l'ipotesi di ottenere l'insieme di tutti i siti web italiani direttamente dalla sorgente, si è ipotizzato di ottenere la lista fornita dalla query $''site : .it''$ richiesta a diversi motori di ricerca; questa query presenta come risultato un insieme di siti web appartenenti al dominio italiano.
Volendo evitare dei provvedimenti negativi da parte dei motori di ricerca, come ad esempio il ban da Google dell'indirizzo IP dal quale viene effettuato il crawling e la rimozione dal servizio di ricerca dei contenuti relativi all'Università di Padova, si è verificato se l'idea di partire dai risultati forniti da un motore di ricerca fosse attuabile.
Esplorando i term of service (TOS) dei più importanti motori di ricerca secondo quanto riportato in [1] e [6] e analizzandoli per popolarità decrescente, appare una situazione così costituita:

- Google: presenta un articolo, il 5.3 che indica come tutti i servizi forniti e tutte le risorse presenti nel sito e i servizi offerti non debbano essere sottoposti al crawling da parte di bot o spider.
  5.3 You agree not to access (or attempt to access) any of the Services by any means other than through the interface that is provided by Google, unless you have been specifically allowed to do so in a separate agreement with Google. You specifically agree not to access (or attempt to access) any of the Services through any automated means (including use of scripts or web crawlers) and shall ensure that you comply with the instructions set out in any robots.txt file present on the Services.

- Msn: come Google anche questo motore di ricerca presenta una particolare ostilità verso software che vogliono analizzare in modo automatizzato i contenuti e i servizi messi a disposizione.
  L'utente si impegna a non: recare danno, disabilitare, sovraccaricare o pregiudicare il servizio o le reti connesse al servizio; utilizzare processi o servizi automatizzati per accedere e/o utilizzare il servizio (come robot,

spider, memorizzazione periodica nella cache di informazioni archiviate da Microsoft o meta-searching).

- Bing: essendo anche questo sito una proprietà Microsoft, verificando il TOS si risulta indirizzati allo stesso documento visualizzato per il sito di Msn, che indica ancora una volta la non possibilità di utilizzare servizi automatizzati per l'analisi di risorse e servizi forniti.

- Yahoo: non sono presenti voci che indicano che non si possa in qualche modo utilizzare un bot o crawler per poter scansionare e analizzare i servizi ed i contenuti del sito web.

- Ask.com: consultando il TOS della parte italiana del sito si nota come non siano possibili usi non personali del motore di ricerca adibiti a scopo lucrativo, collettivo o commerciale.

- AOL search: nei termini d'uso di questo motore di ricerca risulta presente l'articolo 11 che vieta l'utilizzo di servizi automatizzati per accedere al contenuto pubblicato.

Per quanto riguarda i motori di ricerca italiani, considerandoli tra i siti più visitati dagli italiani come suggerito in [3], la situazione è la seguente:

- Libero: il motore non presenta limitazioni nell'uso di servizi automatizzati per la sua analisi.

- Virgilio: questo sito non presenta una voce nelle sue note legali che siano contrarie all'utilizzo da parte di servizi automatizzati come i crawler.

- Tiscali: non è presente un TOS dal quale documentarsi

- Kataweb: non è presente un TOS dal quale documentarsi, ma si può notare come utilizzando il sito per una ricerca appaia il messaggio *"enhancedbyGoogle"*

Pensando quindi ai motori di ricerca come le più fornite e dettagliate collezioni di indirizzi di risorse web ed enti che hanno la più aggiornata versione del web, solamente Yahoo, Virgilio e Libero permettono di dare un contributo significativo al progetto.
Per quanto riguarda Tiscali invece sarà possibile analizzare i contenuti presenti sottomettendosi semplicemente al file robots.txt appartenente al sito, mentre per Kataweb, utilizzando il sistema di Google per la ricerca risulta non sicuro utilizzare il crawler per lo scraping dei risultati.

3.1.3 Caratteristiche di una seed list

Proprio dal contributo di un operatore di Yahoo è stato possibile stilare le caratteristiche necessarie per una buona ed efficiente seed list.
In particolare nel documento [12] è possibile comprendere come la buona qualità di una seed list si possa valutare dal numero di link in uscita di una pagina, che gli stessi link in uscita non siano rivolti verso pagine finalizzate solo a spam e che la pagina stessa inserita nell'insieme iniziale non sia una pagina di spam.
L'importanza degli elementi di una seed list è invece valutata attraverso la popolarità, l'affidabilità e la qualità stessa della pagina utilizzata come elemento iniziale. Maggiore risulta la popolarità, migliore sarà l'analisi che si potrà ricavare come ad esempio numero di risorse analizzate.
La direzione poi verso il quale il crawler dovrà indirizzarsi, sarà sicuramente quella relativa solamente al web italiano, tralasciando ogni via all'interno dei siti web della seed list che si allontani da questo. La risoluzione di questo aspetto è stata considerata successivamente nelle impostazioni del crawler e viene ulteriormente specificata nelle direttive di Heritrix presenti nella seed list.

3.1.4 Creazione della seed list

Da queste premesse sono stati ricercati diversi indirizzi di risorse che potessero, adattandosi ai criteri e alle limitazioni sopra esposte, essere dei buoni candidati per la seed list. Si deve ricordare che il web in generale risulta possedere un grafo non fortemente connesso e quindi non ci sono percorsi che riescono a collegare tutti i nodi della rete.
Nella rete sono presenti delle risorse web chiamate directory che si presentano come delle grandi raccolte di link a siti web, postati da utenti per rendere visibile il proprio sito ai motori di ricerca o perché risultati di ricerche effettuate sul web da enti o associazioni; queste risorse hanno lo scopo di catalogare e raccogliere tutte le risorse presenti nel web. Spesso risulta possibile anche che questi siti al loro interno siano suddivisi per categorie e anche dal punto di vista geografico.
In particolare si è scelto sulla base di opportuni test che se le directory suddividono le risorse web geograficamente e se i risultati appaiono solamente nelle sezioni di esse relative al web italiano, di selezionare solo questo percorso. Ad esempio questa semplificazione è stata apportata nella directory progetto open-source di dmoz una delle più grandi e conosciute directory.
Una prima parte della seed list utilizza quindi un insieme di siti web che offrono servizio di directory, e comprendono al loro interno un insieme di

diversi indirizzi di risorse relative al dominio italiano.
In questa lista viene inserita anche la directory dei motori di ricerca Yahoo, Virgilio e Libero che quindi non vengono interrogati con una query specifica rispondendo con solo un sottoinsieme degli indirizzi di risorse web italiane da essi considerati, ma considerando la directory restituirà la totalità degli indirizzi raccolti nelle istanze di crawling del motore di ricerca stesso.
Sulla base dell'importanza e della popolarità dei siti web da utilizzare come seed list si è voluto considerare il motore di ricerca Google, il più popolare e il più utilizzato secondo gli articoli citati precedentemente, per cercare di ordinare i risultati della query $''site : .it''$ ma non volendo violare il TOS del motore di ricerca i siti web della lista risultato sono stati ottenuti senza utilizzare per questo passo il crawler o altri servizi automatizzati, ma effettuandone una copia manuale. Una seconda parte della seed list utilizza quindi un insieme di circa 890 siti web rilevati dal motore di ricerca Google.

3.1.5 Direttive nella seed list

Sono necessarie per lo scopo del progetto delle direttive di Heritrix da inserire all'interno della seed list anche se non sono degli indirizzi URL. Queste direttive servono per determinare il comportamento del crawler in particolari situazioni e verranno scartate dalla seed list dopo che il comportamento da loro definito è stato impostato nell'istanza di crawling.
Si è reso necessario inserire la stringa $'' + http : //(it,''$ che indica come la ricerca del crawler si possa espandere oltre agli indirizzi iniziali, solamente verso risorse web .it senza dirigersi verso strade esterne a essi.
Risulta anche necessario inserire le direttive di contenimento per indicare al crawler di analizzare nei contenuti presenti nella seed list relativi a pagine non presenti nel dominio italiano solamente la parte utile, fermandosi a siti web e al percorso selezionato appositamente per il progetto senza far uscire il crawler verso altri siti esterni collegati a questi, a meno che non appartengano al dominio italiano. Ad esempio la stringa $''+http : //(org, dmoz, www,)/World/Italiano''$ indica che si vuole far analizzare al crawler non tutte le varie sotto categorie del sito www.dmoz.org ma solamente la sottocategoria italiana indirizzabile dal path così definito.

3.1.6 Possibilità di aggiornamento della seed list

La lista così creata è stata inserita nei settings del job relativo al crawling.
Una volta inserita il software Heritrix permette comunque di modificarla an-

che quando il progetto è avviato e il crawler sta già analizzando le risorse web. Quindi è possibile aggiungere altri elementi o toglierne alcuni attraverso l'interfaccia grafica web oppure modificarla direttamente dal file *"seeds.txt"* presente all'interno della cartella del corrispondente job su disco fisso della macchina.

3.1.7 Riscontri con altri progetti

Per avvalorare la scelta effettuata nell'inserire gli elementi della seed list sono stati ricercati in particolare in portali come [2] e [4] degli articoli scientifici inerenti alla costruzione di un buon insieme di URL di partenza.
Non sono stati ritrovati paper riguardati crawling relativi al web italiano, ma in molte altre ricerche documentate sono stati utilizzati come URL iniziali siti di directory e in particolare *"www.dmoz.org"*. I documenti a cui si fa riferimento sono:

- un progetto di Stanford University documentato in [11] relativo a un progetto di crawling di tutto il web e di come mantenere la freschezza delle informazioni analizzata;

- esempi dell'uso di dmoz come elemento interessante nella lista iniziale per il crawling sono riportati anche in [9] un progetto per valutare brevemente l'efficienza di Information Retrieval basato su un crawling con risultati di svariati elementi;

- uso dello stesso servizio di directory per un crawling focused su siti relativi alla salute e alla medicina come citato in [15], per valutarne qualità e copertura dei siti web ottenuti.

Per quanto riguarda una possibile stima di numero di elementi e possibilità di utilizzare altri tipi di risorse web oltre alle directory sono stati considerati questi documenti:

- una stima della grandezza di una lista iniziale di elementi potrebbe essere definita dall'articolo [8] dove viene descritto un esperimento di focused crawling nel domino .uk e vengono utilizzati circa 190.000 URL iniziali distinti;

- l'articolo di ricerca riportato in [10] risulta invece effettuare il crawling partendo da un insieme dei 50 siti più popolari al mondo per sviluppare un primo approccio sulla ricerca nel web.

3.2 Creazione job di crawling

Un progetto di crawling nel software di Heritrix è chiamato job; esso può essere costruito in modo semplice direttamente dall'interfaccia grafica del software impostando tutti i parametri di avvio, i moduli, i sottomoduli e i settings necessari.
Per le informazioni relative ad un'accurata installazione di Heritrix si rimanda ai comandi riportati in appendice.
Viene presentata ora la configurazione creata e ottimizzata per raggiungere lo scopo del progetto.

3.2.1 Moduli

L'ordine in cui i moduli sono stati inseriti all'interno di ogni classe è importante e determina la buona riuscita delle funzionalità dei moduli stessi. I moduli utilizzati sono i seguenti:

Crawl-scope:
Il primo modulo utilizzato è SurtPrefixScope che risulta essere necessario per lo scopo del progetto, infatti esso permette di analizzare solamente alcune regioni del web ed in particolare con alcune direttive impostate e presenti nella seed list si indica al crawler di analizzare solo il web italiano (con estensione .it) e di non analizzare altri siti esterni a questi se non quelli inseriti nell'insieme di risorse iniziale. L'inserimento del modulo è documentato dalla figura 3.1.

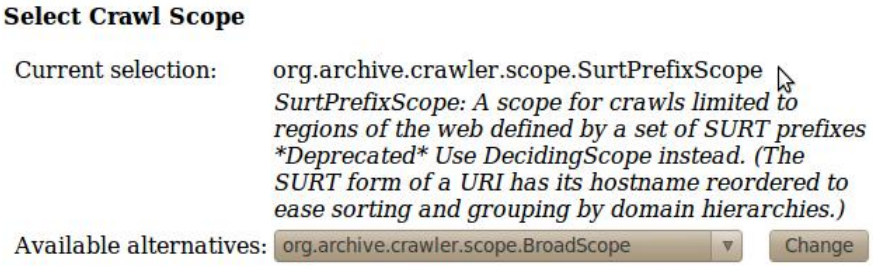

Figura 3.1: Definizione della portata del crawling

Frontiera:
Il modulo designato per il compito di frontiera è BdbFrontier perché non solo risulta essere la frontiera più recente scritta dagli sviluppatori apportando tutti i miglioramenti disponibili, ma permette di effettuare il crawling di un

URI solamente una volta individuandolo come già scansionato nelle sue successive apparizioni e di utilizzare con efficacia la scrittura degli URI su disco permettendo anche di mantenere lo stato interno delle risorse del crawling quindi facilitando il processo di checkpointing dello stato stesso.
BdbFrontier inoltre definisce la visita breatdh-first delle risorse, l'inserimento del modulo è descritto in 3.2.

Preprocessori:
I moduli della categoria processori utilizzati sono solamente quelli di default come mostrato in 3.3.
In particolare Preselector effettua un controllo preventivo sulla risorsa reperita per riuscire a capire se essa è attinente o meno alle condizioni preimpostate nel job di crawling.
PreconditionEnforcer è complementare al precedente poiché assicura che tutti gli URI che passano la sua scansione soddisfano tutte le condizioni e sono quindi validi per future scansioni e analisi più accurate.

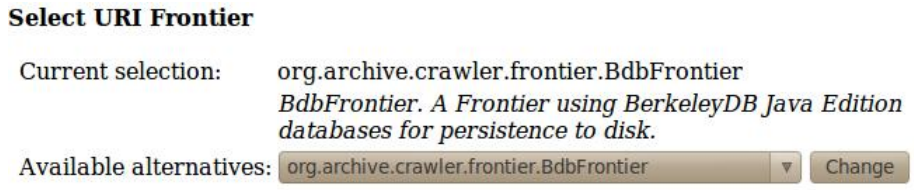

Figura 3.2: Frontiera per la scansione degli URI

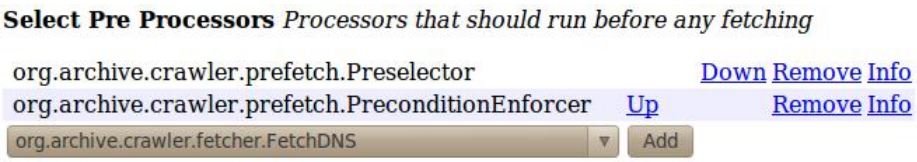

Figura 3.3: Preprocessori per la scansione

Fetchers:
In totale sono tre i moduli che indicano quali protocolli utilizzare per ricavare le risorse nel processo di crawling. Ne vengono utilizzati solamente due in particolare FetchDNS di cui è consigliata l'inclusione perché necessaria per utilizzare il Domain Name System nella risoluzione degli indirizzi IP delle

risorse web, e FetchHTTP utilizzato per reperire tutte le risorse web raggiungibili da un qualsiasi browser web le quali sono l'obiettivo del progetto. I moduli sono stati inseriti come presentato in figura 3.4.
Non viene utilizzato il protocollo FTP per raggiungere le risorse web in quanto non necessarie.

Figura 3.4: Moduli per reperire fisicamente le risorse

Extractors:
I moduli inseriti negli estrattori servono per reperire i link presenti all'interno dei diversi tipi di documenti. In particolare Heritrix permette di estrarli da risorse che utilizzano il protocollo http come pagine web dinamiche, pagine web statiche html, fogli di stile css, script in javascript con estensione js, filmati flash con estensione swf, documenti pdf, documenti xml e documenti di testo scritti da word con estensione doc. Sono stati inseriti tutti i moduli forniti da Heritrix per reperire i nuovi indirizzi per procedere nelle future elaborazioni dalle tipologie di risorse del web che potessero contenerne. L'insieme dei moduli è presentato in figura 3.5.

Writers:
Sono stati eliminati i moduli per la scrittura di file ARC presenti nella configurazione di default perché non necessari al progetto, mentre la scrittura del grafo del web è rimandata al post-processor.

Postprocessor:
Come indicato dall'immagine 3.6 vengono riportati qui moduli come CrawlStateUpdater per effettuare l'aggiornamento delle risorse scansionate nello stato del crawler, LinksScoper per controllare l'effettiva aderenza degli URI estratti e reperiti ma ancora da analizzare alla portata del crawling, FrontierScheduler che si occupa di effettuare lo scheduling di come verranno scansio-

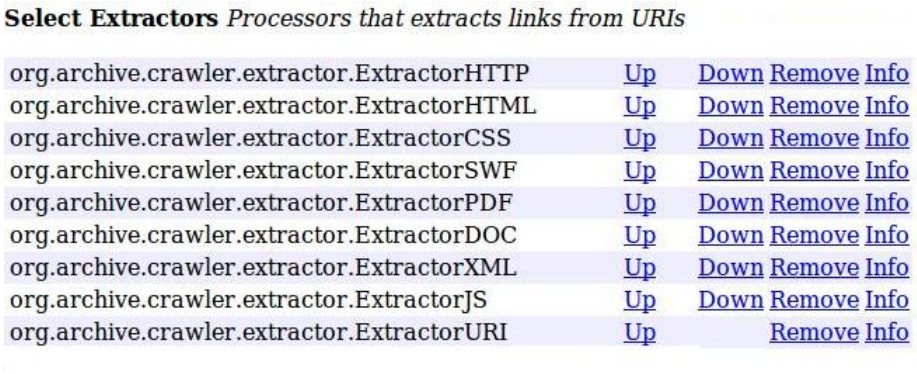

Figura 3.5: Estrattori di link dalle risorse web

nati gli indirizzi delle risorse inserite in coda. Il modulo LowPauseDiskProcessor aumenta le prestazioni di scrittura su disco diminuendo la pausa tra una scrittura e l'altra; GraphWriter risulta essere il modulo scritto appositamente per creare un file di listing degli URI scansionati e un file che vuole essere la scrittura del grafo del web italiano scopo di questo progetto.
Nel file del grafo per ogni nodo padre scansionato viene inserito il suo codice hash MD5 con il bit più significativo posto a uno, seguito successivamente da tutti i nodi figli a esso collegati riportati tutti con il corrispondente codice hash con il bit più significativo posto a 0. SupplementaryLinksScoper è incorporato per ultimo per rafforzare la verifica delle condizioni ed effettuare una seconda analisi sull'appartenenza o meno degli indirizzi delle risorse all'insieme di risorse valide per future analisi del crawler.

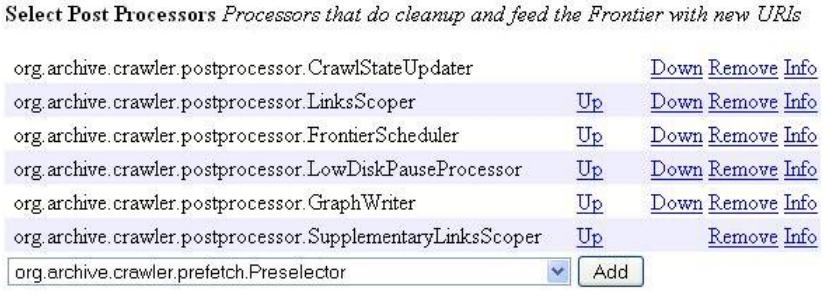

Figura 3.6: Moduli da attivare nel passo di post-processing

Tracking:
Il modulo inserito come da figura 3.7 è il responsabile della creazione di tutte

le statistiche e della barra grafica di avanzamento del crawling disponibile nella pagina principale di Heritrix quando è attivo un job.
Vengono definiti il valore di URI scansionati al secondo e la media di questa grandezza, la velocità di reperimento delle informazioni in KB/s e la sua media, il numero di elementi già scansionati, quello degli elementi in coda e il totale come somma dei due, la tempistica di fine ipotizzata e il tempo passato dall'inizio del crawling. Vengono inoltri monitorati il numero di thread attivi e il rapporto di congestione determinato come numero di URI da scansionare e thread adibiti alla scansione.

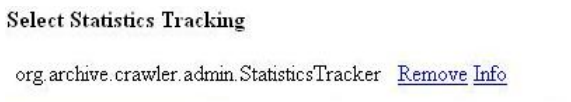

Figura 3.7: Modulo per la creazione di statistiche

3.2.2 Sottomoduli

I sottomoduli servono per configurare il comportamento di tutti i moduli precedentemente importati e della catena di passi iterativi già descritta per l'analisi delle risorse che il crawler riesce a reperire. Ogni sottomodulo può contenere dei filtri o delle opzioni configurabili ma ognuna di esse deve mantenere un nome univoco pena possibili malfunzionamenti dell'istanza di crawling.

Exclude-filter:
Il sottomodulo in figura 3.8 possiede lo scopo di evitare l'analisi del crawler su certi tipi di risorse reperibili nel web, in particolare il filtro utilizzato che possiede il nome univoco escludiImm è di tipo URIRegExpFilter e viene configurato successivamente per indicare quali sono le estensioni delle risorse che si vogliono inibire dall'analisi.

Regole di processo per le risorse
Risulta inoltre essere d'aiuto per lo scopo del progetto l'utilizzo di impostazioni nei sottomoduli di regole per l'analisi delle risorse solamente del dominio italiano.
Queste regole come mostrato nell'immagine 3.9 sono di tipo SurtPrefixedDecideRule e compiono l'azione di applicare il controllo della portata del

Figura 3.8: Sottomodulo per l'esclusione dei file con estensione non desiderata

crawling su tutti i moduli dove vengono attivate in base a direttive particolari inserite nella seed list del crawler. Le suddette regole sono state inserite nei sottomoduli riguardanti i moduli nelle categorie preprocessing, fetcher, preprocessor ed extractor.

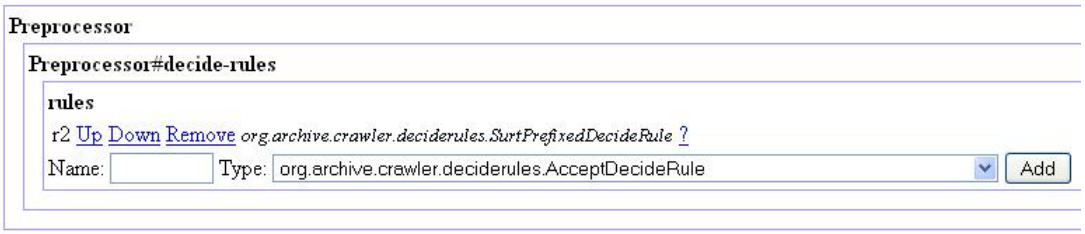

Figura 3.9: Regole per l'analisi delle risorse

3.2.3 Settings

I settings sono impostazioni necessarie a determinare in particolare il comportamento di tutti i moduli e sottomoduli precedentemente selezionati e per definire impostazioni di carattere generale del job di crawling.

Filtro di esclusione
Utilizzando il sottomodulo escludiImm come filtro di esclusione precedentemente presentato, si imposta nei settaggi quali sono le estensioni da poter scartare in fase di analisi del crawler. Le impostazioni sono inserite in figura 3.10. In particolare con la seguente stringa:

```
1 ^.*(?i)\.(bmp|cab|gif|jpe|jpg|jpeg|png|tiff|mid|mp2|mp3|mp4|wav|av\i|mov|↩
    mpeg|ram|rm|smil|wmv|ppt)\$
```

Listing 3.1: Stringa estensioni da escludere

si evita l'analisi di tutte le risorse con estensioni riguardanti le più popolari tipologie di immagini ma anche altri tipi di file come presentazioni, video e audio. Queste risorse non presentano link in uscita ma possono aumentare inutilmente il tempo necessario a effettuare il processo di crawling.

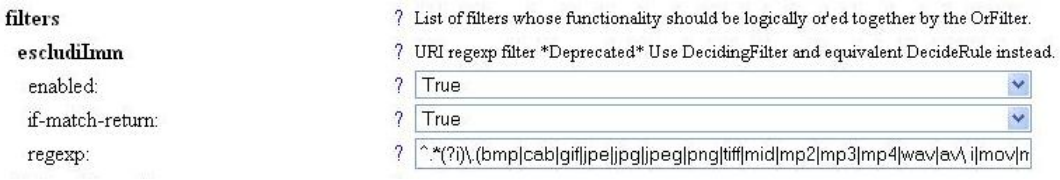

Figura 3.10: Impostazione estensioni da scartare

Impostazioni di riconoscimento

Le due impostazioni presenti in 3.11 da modificare obbligatoriamente riguardano l'identificazione del crawler da parte dei siti web analizzati.
Come mostrato in figura è necessario indicare un sito web raggiungibile da protocollo http nell'user-agent con il quale il software di crawling si identifica a ogni server web che raggiunge nella sua scansione. Osservando l'user-agent si può identificare una prima parte $''Mozilla/5.0(compatible,''$ che indica che si vogliono considerare i contenuti del web reperibili anche dal suddetto browser web, ed una seconda parte $'', heritrix, http://www.unipd.it)''$ che indica il nome del crawler utilizzato e l'indirizzo URL del proprietario del software.
L'indirizzo e-mail configurato nel campo sottostante invece risulta necessario per contattare il proprietario del crawler qualora il software dovesse generare delle anomalie o problemi ai server web con i quali entra in contatto.

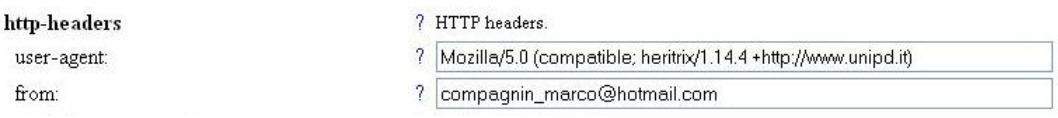

Figura 3.11: Settings per la configurazione http header del crawler

Impostazione numero di toe-threads

Nei settings risulta anche possibile impostare il numero massimo di thread, chiamati in Heritrix toe-threads, che potranno essere utilizzati in modo contemporaneo nel lavoro di crawling del web per l'analisi degli URI.
Nel progetto inizialmente sono stati creati 600 toe-threads massimi come documentato in figura 3.12, ma in caso di avvio o in caso di errori il software gestisce questo numero diminuendoli opportunamente, e l'operatore può modificarli anche mentre il job di crawling è in esecuzione.

Figura 3.12: Impostazione numero massimo di thread contemporanei

3.3 Creazione GraphWriter

Per la realizzazione e la scrittura del grafo del web italiano è stato necessario creare un nuovo modulo customizzato perché la funzione non è presente in Heritrix.
In particolare la scelta di scrivere un modulo della tipologia processore nasce dal fatto che questa classe di moduli dispone di funzioni che vengono eseguite automaticamente una sola volta quando viene avviato il crawling o quando esso viene concluso, e funzioni che vengono automaticamente eseguite a ogni URI analizzato dal crawler.
Proprio quest'ultimo tipo di funzioni possono essere utilizzate per scrivere per ogni uri scansionato la sua parte del grafo del web italiano.
Per l'installazione del nuovo modulo nel crawler si rimanda all'appendice.

```
1  /**
2   * @(#)GraphWriter.java
3   * @author Compagnin Marco
4   * @version 1.00 2010/8/20
5   */
6
7  // aggiunta del processore creato nel package dei post processori
8  package org.archive.crawler.postprocessor;
9  // import delle classi necessarie alla scrittura dei metodi relativi a un ↩
       comune processore
10 import org.archive.crawler.datamodel.CoreAttributeConstants;
11 import org.archive.crawler.framework.Processor;
12 import org.archive.crawler.datamodel.CrawlURI;
13 // import della classe per verificare il match tra due stringhe
14 import java.util.regex.Matcher;
```

```
15 // import delle classi per gli oggetti associati alla scrittura dei file ↩
       su disco per file delle associazioni e file di scrittura del grafo del↩
        web
16 import unipd.util.graph.BinaryFileWriter;
17 import unipd.util.graph.UrlsFileWriter;
18 // import del pacchetto necessario a eseguire operazioni di I/O in java
19 import java.io.*;
```

Listing 3.2: Moduli necessari

Il codice sopra riportato risulta essere la porzione relativa alle direttive di creazione del package e di import delle classi java necessarie al funzionamento.
La prima riga di codice riguarda l'inclusione del modulo in linguaggio java nel pacchetto dei post processori, questa posizione nella catena in prima approssimazione è stata selezionata soprattutto perché è un percorso obbligato da parte degli URI analizzati e perché permette di visualizzare i link uscenti da una pagina attraverso un apposito metodo.
Risultano necessarie le classi che contengono l'interfaccia da implementare per la costruzione del processore CoreAttributeConstants, la classe Processor da estendere per poter utilizzare i metodi comuni a tutti i processori e la classe CrawlURI necessaria per operare con gli URI scansionati dal crawler.
Viene successivamente effettuato l'import della classe Matcher utilizzata per effettuare i confronti tra stringhe relative agli URL scansionati dal crawler.
Per ultime sono state importate le classi necessarie alla scrittura dei file su disco e le classi di default di java per operazioni di I/O.

```
1  /**
2   * Classe per la scrittura del processore customizzato per scrivere il ↩
        grafo del web italiano estendendo la classe processore
3   */
4  public class GraphWriter extends Processor implements ↩
       CoreAttributeConstants {
5
6      // stringa che indica l'espressione da verificare per essere ↩
           considerato un sito del dominio italiano, questo perché nella seed↩
            list ci sono anche siti web esterni al dominio .it
7      final static String REG_IT_DOMAIN = "http://([a-zA-Z0-9]+\\.)+it+";
8      // numero di URI che entrano nel processore di tipo HTTP ↩
           effettivamente analizzati
9      int numberOfCURIsProcessed = 0;
10     // numero di link creati come associazione <pagina padre,pagina ↩
           linkata>
11     int numberOfLinksWritten = 0;
12     // numero di URI scartati perché non aderiscono al dominio .it
13     int numberOfURIsDropped = 0;
14     // oggetto della classe per scrivere il grafo del web
15     BinaryFileWriter file;
16     // oggetto per scrivere l'associazione <pagina,hash MD5>
17     UrlsFileWriter urlsFile;
```

Listing 3.3: Creazione della classe e variabili necessarie

La classe GraphWriter estende il modello di processore creato in Heritrix effettuandone l'overriding per le funzionalità del progetto.
Si utilizza una stringa statica REG_IT_DOMAIN per indicare il formato in cui si dovranno trovare gli URI per essere ritenuti validi, nel dettaglio essa indica il formato standard dell'indirizzo di una risorsa per essere considerata parte del dominio .it, indipendentemente dai sottodomini e dal tipo di cartella o risorsa considerata, questo perché nelle seed list iniziale sono presenti anche nodi che non appartengono al dominio italiano.
Successivamente vengono utilizzate delle variabili intere per conteggiare il numero di URI processati, il numero di link creati e il numero di URI scartati.
Vengono poi dichiarati gli oggetti necessari alla scrittura binaria del grafo del web e per la lista di tutti gli URI scansionati nell'istanza di crawling.

```
1  /**
2   * Costruttore parametrico della classe GraphWriter che imposta i ↩
        parametri visibili dall'interfaccia web su informazioni del modulo
3   * @param nome del modulo visualizzabile dall'interfaccia grafica
4   */
5      public GraphWriter(String name){
6          // il nome e la descrizione verranno visualizzate sulle info del ↩
               modulo
7          super(name, "Un modulo per scrivere le associazioni url hash md5 e↩
                creare il grafo del web");
8          // creazione degli oggetti per associazione url e per grafo ↩
               rispettivamente
9          file=new BinaryFileWriter("./graph/");
10         urlsFile=new UrlsFileWriter("./urls/");
11     }// GraphWriter
```

Listing 3.4: Costruttore della classe

Il costruttore parametrico della classe riceve come parametro il nome del nuovo modulo e viene invocato il costruttore della superclasse Processor per istanziare il modulo una volta avviato il job di crawling.
Vengono poi creati gli oggetti per la scrittura dei file su disco.

```
1  /**
2   * Questo metodo viene invocato dal crawler ogni volta che viene ↩
        scansionato un URI in particolare sarà il responsabile della ↩
        scrittura su disco della lista di URI reperiti e del grafo del web
3   * @param CrawlURI curi rappresenta l'URI correntemente scansionato dal ↩
        crawler
```

```
4   */
5       protected void innerProcess(CrawlURI curi){
6           // controllo se la risorsa analizzata è estratta attraverso ↩
                protocollo HTTP
7           if (!curi.isHttpTransaction())
8               return;
9           // incremento del contatore per ogni URI processato estratto ↩
                utilizzando il protocollo http, quindi vengono scartati le ↩
                informazioni del DNS
10          numberOfCURIsProcessed++;
11          // controllo se la risorsa fa parte del dominio italiano .it
12          String url=curi.toString();
13          String regexp1 = REG_IT_DOMAIN;
14          Matcher match = TextUtils.getMatcher(regexp1,url);
15          // se la risorsa non fa parte della portata del crawling viene ↩
                scartata e  si incrementa il valore delle risorse scartate
16          if(!match.find()){
17              numberOfURIsDropped++;
18              return;
19          }
20      // scrittura della pagina padre nel file della lista di URI analizzati
21          // apertura di un nuovo buffer in formato byte per velocizzare la ↩
                scrittura  su disco e svolgere in un solo accesso per ogni ↩
                analisi
22          byte[] output;
23
24          // estraggo la lista degli URI figli della pagina analizzata
25          int size = curi.getOutCandidates().size();
26          Object[] urlFigli=curi.getOutCandidates().toArray();
27          for(int i=0;i<size;i++){
28              // per ogni URI figlio del padre analizzato mantengo comunque ↩
                    solamente quelli che appartengono al web italiano
29              String figlio = urlFigli[i].toString();
30              match=TextUtils.getMatcher(regexp1,figlio);
31
32              // se esso appartiene allora aggiorno il numero di link ↩
                    costruiti
33              // e viene aggiunto al buffer di byte da scrivere su file
34              if(match.find()){
35                  numberOfLinksWriten++;
36                  // scrittura del suo hash sul buffer output
37              }// if figlio appartiene al dominio italiano
38
39          }//for ciclo di scansione di tutte le pagine figlio dell'URI ↩
                analizzato
40
41          // scrittura del buffer output contenente una parte del grafo del ↩
                web italiano
42          // relativo all'URI analizzato curi nel file del grafo binario su ↩
                disco
43          file.write(output);
44
45
46      }//innerTask metodo invocato per ogni URI analizzato
```

Listing 3.5: Metodo per l'analisi degli URI estratti

Il metodo sopra presentato viene invocato a ogni URI estratto dal crawler, a cui si può accedere dal parametro *CrawlURI curi* passato alla funzione.
Si controlla nel metodo se la risorsa analizzata è stata estratta direttamente

dal protocollo http e quindi non rappresenta una voce DNS necessaria per il funzionamento interno del crawler.
Se la risorsa è effettivamente un indirizzo URL si incrementa il contatore delle risorse analizzate, e se la risorsa analizzata non fa parte del dominio italiano si incrementa il contatore delle risorse scartate.
In caso la risorsa sia obiettivo del progetto e si scrive l'indirizzo della pagina definita pagina padre nel file per la lista delle risorse analizzate e si inserisce la voce nel buffer per costruire il grafo del web.
Si valutano allo stesso modo le risorse figlie della pagina corrispondente, in particolare utilizzando la funzione getOutCandidates() si riesce a risalire a tutte le pagine scovate tra i link uscenti dalla pagina padre analizzata che saranno i candidati delle future scansioni del crawler. La scelta dell'utilizzo di chiamare questo metodo piuttosto che il metodo getOutLinks() risulta influenzata dal comportamento del crawler; il secondo metodo citato infatti restituisce una collezione di URL a cui però sono sottratti le componenti che verranno utilizzate dal crawler perché interne all'insieme delle risorse valide rispetto alla portata del crawling definita nei moduli e sottomoduli. Inoltre il metodo può essere usato solamente all'interno del passo di fetch della catena iterativa di crawling e restituisce una collezione completamente vuota se utilizzata nel passo di post-processing.
La scelta del metodo getOutCandidates() invece restituisce una collezione di elementi alla portata dell'istanza di crawling in esecuzione, e restituisce una collezione non vuota solamente se il metodo viene invocato nel passo di post-processing; queste due caratteristiche sono ideali per il progetto che si è realizzato.
Si effettua un controllo su ogni risorsa figlio per verificare l'appartenenza al dominio italiano dell'indirizzo e in caso affermativo si incrementa il numero di link creati, si scrive l'indirizzo presente nel file della lista e si aggiunge la voce nel buffer che compone la parte del grafo corrispondente.
L'ultima operazione svuota il buffer e scrive la porzione del grafo relativa alle risorse padre e risorse figlie analizzate nel file su disco.

```
1 /**
2  * Questa funzione si occupa di effettuare la reportistica del processore creato, si pensa sia necessario descriverne la funzione e aggiornare i dati della sua elaborazione come numero di URI processati e numero di link scritti su file
3  * @return stringa descrittiva del processore con statistiche su scrittura dei file
4  */
5     public String report() {
6         //le informazioni vengono prima scritte su un buffer e poi scritte sul file del report
```

```
7         StringBuffer ret = new StringBuffer();
8         ret.append("  Processor: org.archive.crawler.postprocessor.↩
              GraphWriter\n");
9         ret.append("  Function:  scrittura del grafo del web italiano \n")↩
              ;
10        ret.append("  CrawlURIs processed: " + numberOfCURIsProcessed + "\↩
              n");
11        ret.append("  Links writen:   " + numberOfLinksWritten + "\n");
12        ret.append("  CrawlURIs dropped:  "+ numberoOfURIsDropped+ "\n\n")↩
              ;
13        return ret.toString();
14    }//report
15 /**
16  * Funzione che viene richiamata una volta soltanto, quando viene ↩
       terminato il crawling si occupa di chiedere gli oggetti per la ↩
       scrittura dei file
17  */
18    public void finalTasks(){
19        // chiusura dell'oggetto per la scrittura del grafo su disco
20        file.close();
21        // chiusura dell'oggetto per la scrittura dell'associazione <URI,↩
              hash>
22        urlsFile.close();
23    }//finalTask
24
25 }//GraphWriter fine della classe
```

Listing 3.6: Metodi di reportistica e chiusura

Il metodo di reportistica permette di far visualizzare i contatori creati come informazioni relative al modulo processore GraphWriter nel file di reportistica di tutti i processori utilizzati dal crawler. Sono di interesse le quantità di risorse analizzate, il numero di link creati, e il numero di risorse scartate perché fuori dalla portata del progetto. Il metodo finalTasks viene eseguito solamente una volta quando il crawling termina, esso è responsabile della chiusura dei file per la scrittura su disco.

Capitolo 4

Scelte di progetto

In questa sezione si presentano i risultati dell'analisi e della ricerca effettuata per la seed list iniziale di URL e per l'adempimento alle politiche di crawling precedentemente menzionate. Seguono questi risultati alcune osservazioni conclusive.

4.1 Seed list del progetto

Dopo aver esposto nei capitoli precedenti l'importanza degli indirizzi iniziali di avvio del crawling ed esposto la ricerca di tali indirizzi si è arrivati a una lista definitiva.
L'insieme di URL di partenza utilizzata nel progetto risulta quella riportata di seguito:

```
1  http://digilander.libero.it/directoryrdm/
2  http://dir.dada.net/
3  http://directory.virgilio.it/
4  http://www.comuni-italiani.it/
5  http://www.dmoz.org/World/Italiano/
6  http://dir.yahoo.com/
7  http://rankdirectory.org/directory/World/Italiano/
8  http://www.abcitaly.com/
9  http://www.allungo.com/
10 http://www.barretta.org/
11 http://www.bloo.it/
12 http://www.bugborg.com/
13 http://www.dica33.net/
14 http://www.directorysi.com/
15 http://www.efind.it/
16 http://www.olx.it/
17 http://www.goldenweb.it/
18 http://www.allwebfree.it/directory.php
19 http://www.harrrdito.it/
20 http://www.iasse.it/cerca.php?tp=directory
21 http://ilmotore.com/
```

```
22 http://www.iltrovatore.it/
23 http://www.iltuosito.it/
24 http://www.infonetfree.com/
25 http://www.initalia.net/
26 http://www.italiatop.com/
27 http://www.metacom.to/
28 http://www.mariorossi.it/
29 http://www.made-in-italy.net/
30 http://www.luxhomepage.it/
31 http://www.laragnatela.com/
32 http://www.freedirectory.it/
33 http://www.kataweb.it/
34 http://www.tiscali.it/
35
36 +http://(it,
37 +http://(com,abcitaly,www,
38 +http://(com,allungo,www,
39 +http://(org,barretta,www,
40 +http://(com,bugborg,www,
41 +http://(net,dica33,www,
42 +http://(com,directorysi,www,
43 +http://(com,ilmotore,www,
44 +http://(com,infonetfree,www,
45 +http://(net,initalia,www,
46 +http://(com,italiatop,www,
47 +http://(to,metacom,www,
48 +http://(net,made-in-italy,www,
49 +http://(com,laragnatela,www,
50 +http://(net,dada,dir,www,
51 +http://(org,dmoz,www,)/World/Italiano
52 +http://(com,yahoo,dir,www,
53 +http://(org,rankdirectory,www,)/directory/World/Italiano
54
55 #insieme degli 890 siti relativi ai risultati di Google rispetto alla ↩
       query "site:.it"
56 #i risultati vengono qui omessi per brevità nell'esposizione
```

Listing 4.1: Seed list per il crawling del web italiano

Nella lista da notare come l'insieme degli indirizzi web relativi alle directory e ai motori di ricerca contattabili, siano stati inseriti per ordine di popolarità decrescente relativamente allo scopo del progetto scelto. Per ultime introdotte dal segno "+" sono inserite tutte le direttive di portata e di limitazione necessarie al progetto.
Si è privilegiato un insieme non troppo esteso di elementi della seed list anche se l'unica stima di grandezza documentata dell'insieme iniziale prevede un gran numero di elementi.
La non estensione non significa però una perdita di qualità della seed list stessa, infatti sono stati inseriti nel breve numero di elementi alcuni URL che rappresentano una fonte di indirizzi molto vasta come le directory e dai quali ci si aspetta un grande contributo come numero di collegamenti uscenti verso risorse italiane.
Questo insieme di elementi risulta essere il definitivo sulla base della ricer-

ca effettuata ed esposta nella sezione precedente ed è stato utilizzato alla partenza dell'istanza di crawling del progetto.

4.1.1 Comportamento rispetto alle risorse dinamiche

La possibile diversità degli URL per una stessa risorsa web dinamica in base all'interrogazione effettuata, come già espresso all'interno della sezione di teoria è stata trattata in questo progetto considerando distinti tutti gli indirizzi dinamici.
In questo modo si possono analizzare singolarmente tutti i link uscenti da una stessa pagina dinamica se essi sono raggiungibili nella scansione del crawler, per considerare anche link che dipendono dalle diverse interrogazioni fatte su una stessa risorsa web.
Questa regola di comportamento non solo facilita il crawling ma permette di ottenere la totalità dei link uscenti da una risorsa web per ottemperare all'obiettivo di reperire la totalità delle pagine web del dominio italiano.
Nel grafo del web questo si traduce in una serie di rami verso tutti i nodi che rappresentano una variante diversa della stessa pagina web ma differiscono per il contenuto dinamico identificato dall'indirizzo della risorsa. Viene quindi rispettata la regola di univocità tra nodo del grafo e URL web della risorsa.

4.1.2 Limite delle pagine protette

Un limite che non risulta superabile da parte del crawler è costituito da quelle pagine che richiedono requisiti di accesso come nome identificativo e password.
Le pagine così protette non possono essere che analizzate per ricavarne i link presenti ma non potranno essere scansionate maggiormente con l'accesso da parte del crawler per un'analisi più approfondita.
Superare questo limite risulta impossibile perché servirebbero le credenziali di accesso a innumerevoli siti web e il crawler è solo un software che compie l'analisi del contenuto delle pagine ma non ha possibilità di ricavare informazioni d'accesso.

4.2 Adempimento delle politiche di crawling

Vengono ora presentati i comportamenti che sono stati specificati attraverso alcune impostazioni del software Heritrix per poter costruire delle politiche di crawling efficaci per il progetto che si vuole realizzare.

4.2.1 Regole per politica di selezione

Rispetto la politica di selezione è stato possibile impostare i seguenti due comportamenti:

- visita breadth-first: la configurazione impostata predilige quindi la visita di tutti gli URL raggiungibili da una stessa risorsa. Questo facilita la costruzione del grafo del web per uno stesso nodo da analizzare scansionando per primi tutti i nodi figli presenti e costruendo gli archi verso essi. Inoltre il tipo di visita è il predefinito per l'inserimento di un modulo necessario alla politica di rivisitazione implementata.

- impostazione della portata del crawler: come documentato precedentemente è stato possibile impostare la portata del progetto di crawling e limitare l'analisi solamente al web italiano attraverso le direttive presenti nella seed list. Si è scelto poi attraverso i moduli citati precedentemente e al codice scritto nella procedura automatizzata per la creazione del grafo, un'analisi limitata solamente al protocollo http.

4.2.2 Regole per politica di rivisitazione

La scrittura del grafo del web necessita sicuramente di mantenere un'associazione univoca tra URL analizzato e nodo del grafo del web.
A questo proposito si è voluto impostare direttamente dal crawler il comportamento da seguire che consiste nella visita univoca di una risorsa web.
Questo comportamento è stato seguito perché risulta più efficiente rispetto a una riduzione per ordinamento e cancellazione degli elementi duplicati presenti nei file del grafo costruiti dalla scansione del crawler. Infatti si ipotizza che una riduzione a posteriori degli elementi duplicati comporti uno spreco del tempo dedicato alla scansione di stessi elementi ripetutamente, mentre risulta più agevole l'analisi univoca a priori degli elementi attraverso un crawling che elimina occorrenze molteplici degli stessi indirizzi web in fase di acquisizione delle risorse.
Il crawler preleva gli indirizzi iniziali uno per volta in sequenza e procede nella loro analisi, quindi configurando una politica di non rivisitazione gli URL iniziali scansionati raggiungeranno molti indirizzi di risorse che potranno essere saltati se venissero raggiunti dall'analisi delle risorse successive.
A supporto di questo comportamento anche l'obiettivo del progetto di ottenere un'immagine del dominio italiano senza rendere necessaria la scansione ripetuta per valutare la freschezza e lo stato di aggiornamento delle pagine web.

Per ottenere la non rivisitazione delle risorse come documentato precedentemente si è configurato il modulo frontiera BdbFrontier.

4.2.3 Regole per la politica di cortesia

La politica di cortesia incide molto sui provvedimenti che un server web può prendere verso un crawler che ne analizza il suo contenuto ed è quindi stato necessario effettuare una visita non troppo invasiva dei siti web analizzati.
In particolare il comportamento scelto per la politica è stato quello di onorare sempre i file *"robots.txt"* presenti in uno spazio web.
Questo comporta come accennato precedentemente la possibilità di non completare l'analisi di tutto un sito web perché potrebbero essere presenti documenti o pagine che vogliono essere tenuti segreti, ma non si sottraggono in questo modo risorse al server web che si vuole scansionare.
Si è posto inoltre un altro vincolo temporale per tutelarsi nel caso il delay di rivisitazione descritto in *"robots.txt"* sia esageratamente alto si è scelto di aspettare un tempo massimo di cinque minuti per la visita di un'altra risorsa dello stesso spazio web.
Le impostazioni sono documentate dall'immagine 4.1 sotto riportata dove viene inserito un comportamento classico rispetto ai citati file.

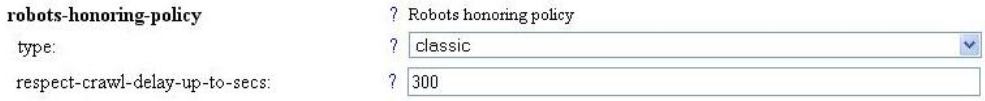

Figura 4.1: Impostazione di aderenza ai file *"robots.txt"*

4.2.4 Regole per politica di parallelismo

Fortunatamente la politica di parallelismo può essere facilmente implementata in Heritrix impostando il numero di toe-thread da utilizzare nelle scansioni.
Si è scelto di passare dai 600 thread inizialmente stimati agli 800 thread contemporanei dopo attenti test mirati su aspetti di consumi di banda, utilizzo di memoria ram da parte della macchina virtuale di Java utilizzata dal software e prestazioni del crawling.
Per quanto riguarda la possibilità di risposta alle problematiche che si possono verificare durante il crawling si è scelto di impostare dei timer relativi alla scansione di una risorsa:

- delay di rivisita: se una risorsa viene contattata e reperita correttamente si è scelto di impostare in modo casuale un tempo di delay compreso tra due e venti secondi per aspettare e provare a ricontattare lo stesso server. Queste impostazioni presentate nella figura 4.2 vengono integrate con il file *"robots.txt"* relativo allo stesso spazio web.

- numero di tentativi: si è scelto inoltre di impostare una quantità di tentativi massima per riscaricare una risorsa che potrebbe essere non raggiungibile pari al trenta impostando una pausa tra un tentativo e l'altro di circa 15 minuti.
  Il grande numero di tentativi è dovuto al fatto che si vuole scansionare la totalità di risorse che si possono reperire e quindi provare a reperire anche le risorse che non vengono reperite istantaneamente. Il grande delay risulta necessario perché se l'errore di scansione è dovuto alla rete si possa arrivare a una soluzione. Nella figura 4.3 sono presenti le impostazioni attuali.

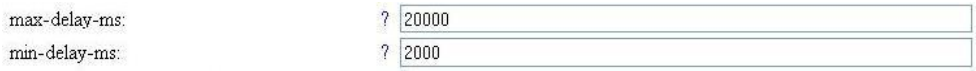

Figura 4.2: Estremi di delay per risorse su uno stesso server

Figura 4.3: Numero di tentativi e delay tra due tentativi successivi

4.3 Osservazioni sulle scelte

Il progetto di crawling si basa in gran parte sulla seed list appositamente studiata ma non c'è la certezza assoluta che questa possa portare alla totalità delle risorse del web italiano anche se lo scopo è stato così definito.
Non si riesce nemmeno a stabilire quante possano essere in totale e in modo certo le pagine relative al dominio italiano, ma si ha la certezza solo sul numero di domini registrati presso l'autorità competente.
Anche se la lista iniziale di URL riportata risulta essere definitiva si può operare in modo correttivo a posteriori per poter cercare di aumentare il numero di risorse analizzate. In particolare si può effettuare un ripristino del

crawling dall'ultimo traguardo salvato e successivamente modificare aggiungendo nuovi URL la seed list che possano contenere un gran numero di link in uscita e rispettare comunque le indicazioni di qualità descritte nei paragrafi precedenti.
Un'operazione di questo tipo non conoscendo il numero di elementi totali da dover raggiungere deve essere valutata attentamente in base alle stime effettuate a priori sul numero di risorse che si ipotizza siano presenti.
Oltre all'insieme di indirizzi iniziali lo studio fin qui condotto ha portato a tutte le configurazioni e i passi necessari per svolgere un crawling ben strutturato per realizzare gli scopi prefissati del progetto.

Conclusioni

Il lavoro descritto in questa relazione si colloca all'interno di un progetto che ha come finalità il crawling esaustivo del web italiano e la costruzione del grafo associato. Abbiamo realizzato una ricerca per la stesura di una lista iniziale di elementi, dalla quale poter ipotizzare di raggiungere tutte le risorse italiane presenti nel web. In seguito abbiamo configurato e gestito il processo di crawling.
La lista definitiva di elementi da cui il crawling ha inizio è composta da un insieme selezionato di directory web come dmoz che contengono una lista di link a risorse del dominio italiano raccolta da altri progetti di crawling o da segnalazioni degli utenti del web. Considerando i termini di servizio presenti nei motori di ricerca, solo una parte di quelli consultati tra cui Yahoo, Virgilio e Libero sono stati utilizzati per dare un contributo al progetto. Quindi la lista iniziale è stata completata con risorse come directory accessibili dei motori di ricerca e URL ottenuti dalla particolare interrogazione $''site : .it''$.
Il processo di crawling è stato sviluppato implementando le politiche di selezione, rivisitazione, cortesia e parallelismo. Per la scrittura del grafo del web è stato appositamente creato un modulo specifico per memorizzare gli indirizzi URL scansionati dal crawler e scriverne le relazioni di tipo padre-figlio tra gli stessi.

Appendice A

Complementi all'installazione di Heritrix

In questa parte dell'appendice vengono sviluppati le informazioni complementari per una corretta installazione e avvio del software di crawling e dei suoi moduli.

A.1 Avvio del software

Per un corretto avvio di Heritrix attraverso terminale devono essere inseriti i seguenti comandi:

```
1 export HERITRIX_HOME=/home/user/heritrix
2 cd $HERITRIX_HOME
3 chmod u+x $HERITRIX_HOME/bin/heritrix
4 JAVA_OPTS="-Xmx3072m" \$HERITRIX_HOME/bin/heritrix --bind / --admin=XXXX:↩
      YYYY
```

Listing A.1: Script di avvio Heritrix

Viene prima definita come variabile d'ambiente la directory nella quale sono stati installati i file di Heritrix attraverso il comando export della shell di linux.
Nella seconda riga di codice semplicemente si accede alla cartella di lavoro prima definita.
Nella terza riga attraverso l'assegnazione dei privilegi utente al processo si permette invece l'esecuzione del file heritrix.jar, cuore del software di crawling dove sono contenuti tutti i file dei moduli da utilizzare.
La quarta riga descrive tre impostazioni particolarmente importanti: la prima imposta il valore di memoria ram utilizzabile dal crawling in MB e la assegna

al processo del crawling; la seconda impostazione è relativa all'avvio del software su tutte le interfacce di rete disponibili nella macchina in uso; la terza impostazione configura un utente admin per l'accesso da interfaccia web protetto attraverso autentificazione con parametri quali nome utente - XXXX e password YYYY impostati.
Una volta eseguiti questi comandi diventa disponibile l'interfaccia grafica raggiungibile da un qualsiasi browser dove poter interagire con il crawler. Se non viene modificata l'impostazione nel file heritrix.options, la porta di default su cui è attivo il servizio è la 8080.
L'interfaccia grafica presenta la metodologia principale per la creazione di istanze di crawling chiamate job, ognuno identificato da un nome e un codice univoco assegnato automaticamente.

Dopo essersi autentificati e seguito il link dalla pagina web principale di

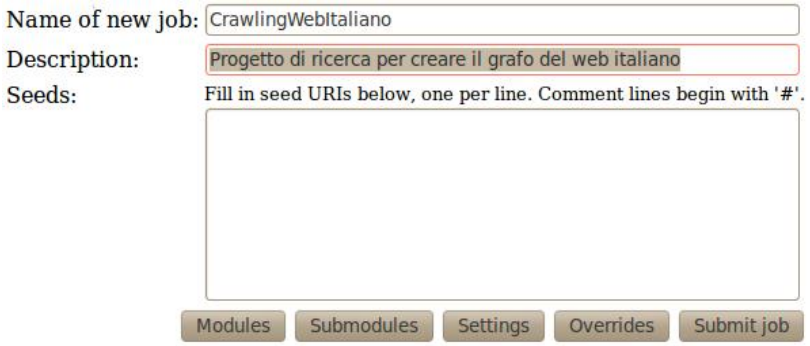

Figura A.1: Creazione nuovo job

Heritrix per la creazione di un nuovo job di crawling è possibile definirne il nome, una semplice descrizione e la lista di URL iniziale da cui effettuare il crawling come mostrato in figura A.1.

A.2 Segnalazione del modulo

Una volta creato il modulo per la scrittura del grafo del web italiano, risulta necessario segnalare al software Heritrix il percorso dove è presente il modulo nella memoria dell'elaboratore utilizzato, il suo nome e di che categoria fa parte.
Queste operazioni si compiono brevemente compilando il modulo java creato e successivamente inserendo GraphWriter.class nel percorso $''/org/archive/crawler/postprocessor/''$ all'interno dell'archivio jar core di Heritrix poiché

questo è stato il path segnalato nella direttiva package all'interno del modulo creato.
Per segnalare nome e posizione del nuovo modulo al crawler si deve modificare il file Processor.options all'interno della cartella $''/modules/''$ del jar di Heritrix inserendo la stringa:

```
1 org.archive.crawler.postprocessor.GraphWriter|GraphWriter
```

Listing A.2: Inclusione GraphWriter nei moduli di Heritrix

In questo modo la posizione del nuovo modulo è stata segnalata al crawler e risulta ora possibile utilizzare GraphWriter importandolo come modulo in un qualsiasi job creato in Heritrix direttamente dall'interfaccia grafica come si è precedentemente fatto per qualsiasi modulo di default già presente all'interno del file core heritrix.jar.

Bibliografia

[1] http://comscore.com/. sito web, ultima visita: Agosto 2010. informazioni relative all'uso dei motori di ricerca internazionali.

[2] http://portal.acm.org. sito web, ultima visita: Agosto 2010.

[3] http://www.alexa.com/topsites/countries/it. sito web, ultima visita: Agosto 2010. classifica dei cento siti italiani più visitati.

[4] http://www.ieee.org. sito web, ultima visita: Agosto 2010.

[5] http://www.registro.it. sito web, ultima visita: Agosto 2010. top level domain relativamente al dominio .it.

[6] http://www.seoconsultants.com/search-engines/. sito web, ultima visita: Agosto 2010. informazioni relative all'uso dei motori di ricerca internazionali.

[7] F. M. A. Broder, R. Kumar. Graph structure in the web. *Computer Networks*, pages 309–320, 2000.

[8] S. V. Alessio Orlandi. Compressed collection for simulated crawling. *SIGIR Forum*, pages 39–44, 2008.

[9] G. Bordogna. Information retrieval distribuito e sul web. *http://www.unibg.it/dati/corsi/9007/*, 2010.

[10] C. J. e. C. P. Carlos Jensen, Chandan Sarkar. Tracking website data-collection and privacy practices with the iwatch web crawler. In *Proc. of SOUPS*, pages 29–40, 2007.

[11] J. Cho. *Crawling the web: discovery and maintenance of large-scale web data*. PhD thesis, Stanford University, 2002.

[12] P. Dmitriev. Host-based seed selection algorithm for web crawlers. US Patent App. 20,100/114,858, 2008.

[13] K. M. S. Dill, R. Kumar and A. Tomkins. Self-similarity on the web. In *Proc. of ACM VLDB*, pages 300–329, 2002.

[14] A. S. Tanenbaum. Reti di calcolatori. Pearson Prantice Hall, 2008.

[15] C. e. G. Tin Tang, Hawking. Focused crawling for both topical relevance and quality of medical information. In *Proc. of ACM CIKM*, pages 147–154, 2005.