UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche, Informatiche e Matematiche

CORSO DI LAUREA IN INFORMATICA

Strumenti per l'esplorazione efficiente della letteratura scientifica

Relatore:

Dott. Mauro Andreolini

Laureando:

Paolo Crotti

Anno Accademico 2018/2019

Sommario

1 Introduzione

La produzione scientifica si estrinseca attraverso la pubblicazione di saggi innovativi (nel seguito, *articoli scientifici* o *articoli*) volti da avanzare l'attuale livello di conoscenza umana su specifici argomenti. Gli articoli scientifici sono solitamente scritti seguendo il metodo di indagine *scientifico* che permette, attraverso una precisa serie di stadi, di giungere a conclusioni solide.

Attraverso l'osservazione attenta del fenomeno, vengono formulate delle ipotesi che andranno poi verificate tramite prove e misurazioni; questo permette di giungere a risultati certi e riconosciuti dalla comunità scientifica, perché supportati da esperimenti ripetibili; l'analisi dei dati riveste quindi un ruolo fondamentale in quanto permette di ricavare informazioni utili da nozioni grezze. Questo processo inizia con la raccolta dei dati e la loro organizzazione all'interno di strutture che ne facilitino le successive operazioni; ne segue la pulizia che ha lo scopo di eliminare doppioni, outlier ed eventuali errori nelle misurazioni (come il rumore). Segue l'analisi vera e propria che, tramite l'ausilio di algoritmi, permette di estrarre le informazioni. Non meno importante è la visualizzazione dei dati e dei risultati che ha come scopo quello di presentare in modo chiaro ed efficiente le conclusioni raggiunte. L'importanza di questo aspetto della ricerca è data dal fatto che le pubblicazioni saranno valutate da colleghi accademici ed esperti del settore con l'obbiettivo di giudicare l'effettivo contributo e la forza innovativa delle idee proposte. Esistono vari strumenti che permettono di raggiungere questo obbiettivo, un esempio possono essere i grafici e le tabelle e la scelta sarà sempre legata, non solo alla struttura delle informazioni che si vogliono rappresentare, ma anche al tipo di pubblico a cui ci si vuole rivolgere.

La scrittura di un articolo comporta delle difficoltà: è necessario definire l'obbiettivo e le tappe per raggiungerlo, suddividere il corpo in paragrafi chiari e intuitivi, fornire una moltitudine di esempi e similitudini così da favorire la comprensione da parte del lettore, ma soprattutto richiede una visione completa dell'argomento di cui si sta parlando (1). La ricerca scientifica viene svolta su un argomento ben preciso attorno al quale si possono individuare delle problematiche

ben chiare; inoltre la ricerca deve poter aggiungere novità oppure rivedere in una diversa ottica argomenti già affrontati. Il suo scopo è quello di essere utile agli altri aggiungendo qualcosa di nuovo e tutti gli altri lavori fatti sullo stesso argomento dovranno tenerne conto. La ricerca deve inoltre fornire elementi che permettano di metterla in discussione (verifica delle conclusioni raggiunte) e consentire ad altri di continuarne il lavoro.

Per raggiungere tutti questi obbiettivi è necessario scrivere articoli di qualità ed è per cui fondamentale informarsi approfonditamente sull'argomento, prima di iniziare il lavoro. La prima legge della comunicazione di Wittington dice che "quando qualcuno spiega un argomento che non ha ben capito, sarà compreso solo da chi ne sa più di lui" (2). Quindi se si scrive un articolo senza aver compreso appieno l'argomento di cui si sta parlando, è probabile che il lettore non riuscirà a capirlo. Ne consegue che avere una profonda conoscenza dell'argomento aiuta a:

1) Non commettere errori che comprometterebbero l'immagine dell'autore e dell'editore.
2) Scegliere gli aspetti più interessanti da trattare, individuati grazie all'esperienza personale maturata col tempo.
3) In generale a scrivere meglio: usare gli esempi e le parole più efficaci e creare un filo logico più chiaro.

Per raggiungere questo obbiettivo è cruciale la realizzazione della sezione denominata *related work* (*stato dell'arte*), definita come "il punto cui sono arrivate le ricerche in una determinata disciplina" (3) . Tale sezione dettaglia il lavoro svolto nell'ambito della ricerca da altri colleghi, illustra come il lavoro proposto si differenzi dagli approcci preesistenti e costituisce, in ultima analisi, la motivazione stessa dei contributi dell'articolo. Purtroppo, ai giorni d'oggi, produrre una sezione completa sullo stato dell'arte è un lavoro sempre più arduo perché per avere una visione completa dell'argomento sarebbe necessario che l'autore fosse a conoscenza di qualsiasi pubblicazione mai uscita al riguardo ed è impossibile per un essere umano revisionarla manualmente nella sua interezza. Nel periodo che va dalla fine degli anni '90 al 2015, il numero delle pubblicazioni è aumentato in modo esponenziale, in particolare modo negli ambienti di carattere scientifico/matematico come l'economia, la computer science e la medicina. (4) Negli ultimi anni si è vista

una rapida espansione di articoli legati ad argomenti considerati di nicchia. Ne è un esempio il machine learning: branca della computer science che fa largo uso della statistica nata nei primi anni 70 ma che ha iniziato a diffondersi solo verso la fine dei 90. Questa crescita esponenziale è stata possibile grazie al progredire della tecnologia che ha dotato i computer di una maggiore potenza di calcolo e dalla possibilità di accedere, in modo semplice, a una più vasta mole di informazioni (sviluppo e diffusione del Web).

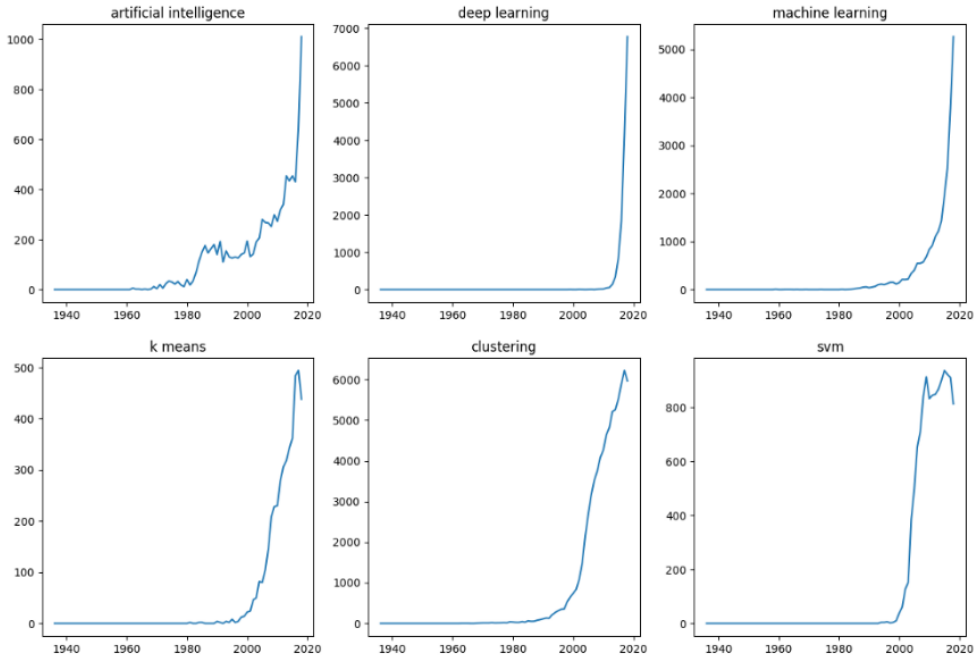


FIG. 1: Crescita di topic legati al Machine Learning nelle pubblicazioni scientifiche

Si può vedere come la diffusione di questa nuova disciplina abbia portato con sé la crescita di molti rami legati a essa (come il k-means e svm), ma anche di campi di studio paralleli a lei, come il deep learning. Si può quindi notare come lo sviluppo in un campo specifico porti la comunità scientifica ad esplorarne i vari aspetti, fare ricerca su di essi e produrre quindi pubblicazioni in cui si annunciano le scoperte effettuate. Questo porterà altri ricercatori ad approfondire, testare, confutare queste conclusioni; porterà a proporre nuovi approcci o ad utilizzare le conoscenze raggiunte in ambiti diversi, a cui non si era pensato. Si vede quindi con quale rapidità e estensione si propaghi questo sviluppo e insieme a lui anche il numero delle pubblicazioni.

Altro fattore che ha favorito la crescita del numero delle pubblicazioni è stata la nascita di siti specializzati nel trovare, indicizzare e mostrare pubblicazioni; dei veri e propri motori di ricerca (*search engine*) per articoli scientifici. Ne esistono di vari tipi: alcuni conservano articoli relativi solo ad argomenti specifici come PubMed che è il punto di riferimento per quanto riguarda la letteratura scientifica biomedica; altri invece trattano tutti i campi di studio, un esempio è Google Scholar che cerca nel Web qualsiasi tipo di articolo (anche non di carattere scientifico). Tuttavia, tali motori di ricerca mostrano alcuni limiti come la libertà che viene data all'utente quando esegue una ricerca. Pochi motori di ricerca, in particolare quelli che trattano ambiti specifici, come PubMed, permettono di formulare query in diversi formati; l'utente può infatti decidere tra inserire delle parole chiave da cercare (sistema classico) oppure può, tramite la ricerca avanzata, specificare il valore dei metadati degli articoli, metadati che possono essere molto specifici, come la paginazione dello stesso. In generale la ricerca avanzata permette di specificare una frase, dove cercare le parole (solo nel titolo, solo nel testo o in entrambi), l'autore, il giornale che lo ha pubblicato e la data in cui è stato scritto. Alcuni sono accessibili solo tramite abbonamento come Web of Science o permettono una maggiore accesso ed analisi della propria base di dati dietro sottoscrizione, un esempio sono le API Academic Knowledge di Microsoft Academic che offrono servizi come il completamento automatico delle query e una migliore visualizzazione dei risultati, tramite grafici e tabelle che mettono in relazione i collegamenti tra i dati. Altri limiti possono essere la lentezza nell'eseguire le query, quindi il lasso di tempo che trascorre dal momento in cui premo invio e quando mi vengono mostrati i risultati (problema che affligge particolarmente Microsoft Academic). Oppure il numero limitato di richieste che soddisfano in un certo lasso di tempo, Scholar per esempio, dopo che l'utente ha trascorso del tempo a navigare nel sito gli può chiedere il controllo reCAPTCHA, per verificare che sia effettivamente un essere umano.

Un limite che accumuna tutti i search engine, almeno nella loro versione gratuita, è l'efficacia nell'illustrare la totalità dei lavori proposti. Infatti, si limitano spesso a mostrare un elenco sequenziale dei lavori proposti, tipicamente ordinati per rilevanza. Molti permettono un diverso tipo di ordinamento, per data per esempio;

altri permettono anche di effettuare altre operazioni di filtraggio dopo l'esecuzione della query.

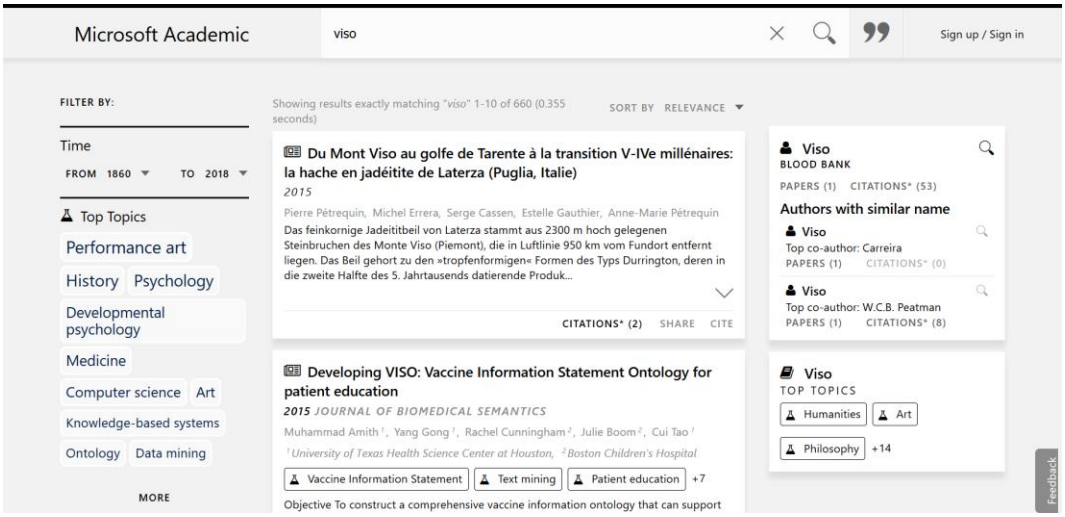


FIG. 2: Modalità di ricerca avanzata fornita da Microsoft Academic

Per esempio, Microsoft Academic permette di specificare un range di date entro il quale deve cadere la pubblicazione, oppure uno o più autori tra quelli che sono comparsi più spesso tra i risultati. Questo rende difficile dedurre dal volume di articoli prodotti gli argomenti principali e le linee di ricerca intraprese nel tempo. Alcuni motori di ricerca riescono ad estrapolare argomenti di interesse legati al contenuto della ricerca ma non riescono a presentarli in maniera organica e strutturata.

Per ovviare a tali inconvenienti, vengono proposti una serie di strumenti che guidino il ricercatore ad informarsi in modo completo. Vengono proposte classifiche che mostrino i campi di studio o gli autori più importanti, nonché un metodo innovativo di sintesi dello stato dell'arte che mira alla creazione di un grafo. In questo grafo, i nodi sono gli articoli mentre gli archi rappresentano le citazioni; l'idea è quella di partire da una o più pubblicazioni, raccoglierne gli articoli correlati e costruire dei grafi da cui estrarre informazioni, come gli articoli e gli autori più importanti per un certo insieme di parole chiave o in base ai temi più diffusi. L'obbiettivo con cui viene costruito questo grafo è quello di mettere in evidenza le relazioni tra articoli scientifici del passato degni di nota e linee di ricerca passate ed attuali, in maniera gerarchica. La costruzione del grafo prevede diversi passaggi non banali, quali ad esempio l'estrazione di articoli passati giudicati rilevanti, il recupero dei metadati

di un articolo necessari alla produzione del grafo quali autori, titolo, date, articoli citati, articoli citanti; nonché l'organizzazione delle informazioni in una base di dati adatta al formato delle stesse, in questo caso un database basato sui grafi. L'insieme di questi passi costituisce una *ricerca* che quindi comprende tutto il processo che, partendo da un documento iniziale, porta ad esplorare quelli che lo citano e sceglierne alcuni in base a uno specifico criterio; per ciascuno di essi si ripeterà il processo di esplorazione fino al raggiungimento di una qualche soglia.

L'approccio proposto è stato implementato tramite un software scritto nel linguaggio Go. Il software è rilasciato con licenza MIT e disponibile su piattaforma GitHub (5), Selenium è rilasciato sotto licenza Apache 2.0, mentre la community edition di Neo4j con GLP v3.

La tesi è organizzata nel modo seguente: nel terzo capitolo viene illustrata una panoramica delle fasi procedimento: verranno quindi discusse le fasi di raccolta delle informazioni dai motori di ricerca, la loro memorizzazione su supporti adatti alla loro struttura e il linguaggio di programmazione. Nel quarto verranno presentati in modo approfondito gli strumenti utilizzati e le loro caratteristiche. Nel quinto sono presenti le varie funzionalità offerte dal software e i servizi che sono in grado di offrire che verranno poi valutati nel sesto capitolo; che conterrà anche osservazioni sui risultati. Nel settimo e ultimo capitolo vengono presentate le conclusioni raggiunte tramite il lavoro di ricerca ed alcuni possibili sviluppi futuri.

2 Progetto

In questa sezione presenterò un'analisi degli strumenti e delle funzionalità necessarie per la creazione del grafo dello stato dell'arte e di altri strumenti utili per l'analisi della letteratura scientifica.

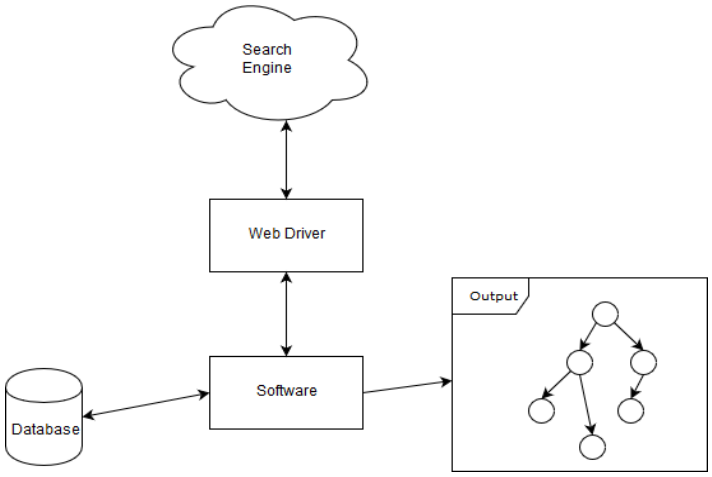


FIG. 3: Schema dei componenti del progetto

2.1 Motori di ricerca

I motori di ricerca specializzati nel trattare la letteratura accademica hanno reso la ricerca delle informazioni, che si svolge solitamente durante un lavoro di ricerca, molto più semplice e veloce. Non è quindi più necessario visitare svariati siti alla ricerca di articoli che trattino l'argomento su cui si sta lavorando, con il rischio di navigare per ore su anche una decina di siti diversi per poi lasciarsi sfuggire degli articoli interessanti. I search engine scandagliano migliaia di siti web che ospitano pubblicazioni scientifiche (come IEEE e Elsevier) e grazie al lavoro di indicizzazione eseguito sulle pubblicazioni trovate, permette una ricerca ad alte prestazioni che facilita notevolmente il lavoro del ricercatore. Non tutti i motori di ricerca trattano tutta la letteratura accademica; molti infatti sono realizzati dalle stesse associazioni che raccolgono le pubblicazioni (*library*) e che quindi permettono una ricerca limitata alla propria base di conoscenza. Un esempio è IEEE

Explore che permette di cercare e leggere articoli, verbali di conferenze, resoconti tecnici e materiale collegato principalmente alla computer science, ingegneria elettrica e meccatronica; contiene infatti materiale pubblicato principalmente da "Institute of Electrical and Electronics Engineers" (IEEE) ed altri partner. Nonostante permetta di cercare solo articoli presenti nella base di dati di IEEE, questo search engine dispone di più di 4,5 milioni di documenti, copre circa 200 giornali e ha a disposizione più di 2.400 libri. In totale copre poco meno del 30% di tutta la letteratura scientifica che riguarda la computer science. (6) Esistono anche altri esempi come ASCE Library che fornisce in formato full-text un database che conserva tutti gli articoli e i documenti pubblicati dalla "American Society of Civil Engineers"; oppure PubPsych che offre libero accesso a articoli di carattere psicologico, sempre in formato full-text.

Tutti però presentano lo stesso limite: permettono di cercare solo all'interno di una cerchia molto ristretta, sia per quanto riguarda la quantità di pubblicazioni, ma soprattutto per gli argomenti che vengono trattati. IEEE Explorer possiede un discreto numero di articoli e di giornali ma è limitato a un ambito di ricerca molto ristretto, non è infatti possibile trovarci articoli di carattere medico o filosofico. Si è quindi scelto di utilizzare come strumento di ricerca per gli articoli, search engine che trattassero in modo più ampio e completo la letteratura accademica. La scelta è ricaduta su Google Scholar, sviluppato dalla Google e uno dei più usati e popolari al mondo e Microsoft Academic, più recente di Scholar ma fornisce molti metadati sugli articoli e dispone di una notevole base di ricerca. Altra caratteristica che ha favorito la scelta di questi due search engines è il fatto che siano gratuiti, non è quindi necessario sottoscrivere un abbonamento o essere affiliati a una particolare società. Diverso è il caso Anthropological Literature, un database che consente la consultazione online ai documenti conservati nella biblioteca di antropologia dell'università di Harvard. L'accesso a questi documenti è disponibile gratuitamente solo all'interno della facoltà di Harvard, ai membri dello staff e agli studenti; per tutti gli altri è necessaria una forma di iscrizione a questo servizio.

Una caratteristica che accumuna Scholar e Academic è il fatto di fornire solo la funzione di ricerca. Mentre molti altri search engine, in particolare quelli legati a un editore o una società specifica, mantengono un proprio database con gli articoli

(in formato full-text) e tutti i loro metadati, come le informazioni sull'autore, la data e il numero di pagine; questi due motori di ricerca si limitano ad ispezionare pagine web alla ricerca di articoli, quando ne trovano uno ne estraggono le informazioni disponibili (quelle che il gestore del sito decide di mettere a disposizione degli utenti), le memorizza nella propria base di conoscenza e le indicizza per velocizzare la fase di ricerca. Ne consegue che nei risultati che vengono restituiti all'utente ci sarà un link alla pagina che ospita l'articolo, non il suo testo; pagina che può richiedere un qualche tipo di autenticazione prima di permettere all'utente di leggere il documento. Tuttavia, entrambi i search engines sopperiscono a questo problema: quello di Google fornisce spesso, di fianco al risultato, uno o più link a pagine web in cui l'articolo è fruibile liberamente; anche in formato PDF. Dal canto suo, Academic divide tutte le fonti da cui è possibile reperire l'articolo in due gruppi: nel primo la risorsa è disponibile in formato html, nel secondo in formato PDF. Per entrambi (in particolare per il motore di ricerca di Microsoft) è quasi sempre possibile risalire al contenuto dell'articolo; è bene però notare che i siti potrebbero ospitare le pubblicazioni in forma gratuita senza il permesso dell'autore o dell'editore.

2.2 Web Scraping

Una volta decisi i motori di ricerca da cui si sarebbero ricavati gli articoli, è stato necessario decidere come recuperare le informazioni su di essi. Questa procedura viene definita *web scraping* e consiste nel estrarre dati da siti web. Il software accede alle pagine tramite un browser o direttamente mediante protocollo Hypertext Transfer Protocol (http) e in modo automatico ne estrae i dati ritenuti importanti; tipicamente questi dati verranno salvati in un database per una successiva analisi. Questo processo consta di due fasi: *fetching* e *extracting*. La fase di fetching consiste nel download della pagina per poi procedere all'estrazione delle informazioni. Possono essere effettuate diverse operazioni sul contenuto di una pagina: *parsing*, ricerca di parole chiave e la formattazione del testo; questo permette di facilitare il recupero delle informazioni. Alcuni esempi sono il *contact scraping* il cui scopo è quello di trovare e copiare nomi, numeri di telefono e URL

in giro per il web e il *product review scraping* che viene utilizzato per monitorare le variazioni di prezzo su specifici prodotti, per tenere d'occhio la concorrenza nel mercato online. (7)

Fenomeno simile allo scraping è il *crawling,* metodo usato dai motori di ricerca come Google per trovare e successivamente indicizzare le pagine web. In questo caso, uno strumento chiamato *crawler* (o *spider*) ha il compito di navigare per il web alla ricerca di nuove pagine da indicizzare; tipicamente viaggia tra i link che trova nelle pagine che visita. I siti pubblici che non desiderano essere ispezionati possono, in un file apposito (robots.txt), specificare quali zone del loro sito il crawler è libero di indicizzare; tuttavia queste sono semplici indicazioni e sta all'agente decidere se rispettarle o meno. La differenza principale tra questi due sistemi di esplorazione è il loro scopo: il crawling ha come obbiettivo la creazione di un indice delle pagine visitate, per permettere poi la ricerca di parole chiave; lo scraping si concentra solo sulla raccolta e successiva analisi delle informazioni. Altre caratteristiche tipiche dello scraping sono: ha un obbiettivo/dominio molto specifico (per esempio può cercare solo all'interno di un sito web), non si cura dei limiti chiesti dal proprietario del sito (robots.txt), agisce tramite un browser, può immettere dei dati nei form e fare il submit ma soprattutto eseguire codice Javascript. Lo scraping è quindi una componente del crawling, in quanto lo spider ha bisogno di analizzare il contenuto della pagina per capire quali siano i link presenti e dove spostarsi successivamente; nonché per reperire alcune informazioni che di solito forniscono i search engine come il titolo della pagina o gli *snippet*, cioè poche righe di testo che descrivono il contenuto della pagina. (8)

Esistono vari strumenti che permettono di fare scraping, in generale qualsiasi strumento permetta di scaricare il contenuto html di una pagina è uno scraper. Le pagine che contengono i risultati sono spesso complesse in quanto sono molto voluminose (molte righe di codice html da controllare) e ricche di nomi di classi (degli elementi della pagina) non proprio intuitivi. L'analisi che meglio permette di gestire una situazione del genere è la *DOM parsing* in cui viene fatto il parsing del contenuto della pagina per poi inserirlo in un albero DOM, la cui struttura dipende dall'applicazione che ha effettuato il parsing. Questo sistema ha molteplici vantaggi: innanzi tutto permette di accedere alle informazioni con estrema facilità

e velocità, in quanto è spesso l'applicativo che gestisce l'albero a fornire funzioni che ne permettono l'accesso, tipicamente specificando il percorso (lista di tag) necessario per raggiungere l'elemento o una sua qualche proprietà. Altro vantaggio è dato dal fatto che sfrutta un browser per navigare/scaricare la pagina web, questo permette al programma di ottenere il contenuto generato dinamicamente dagli script client-side.

Per poter usufruire di tutte queste funzionalità è stato necessario appoggiarsi a un *web driver,* uno strumento che permette di navigare sul web in modo automatico. Programmi come curl o wget che permettono di avere accesso e scaricare il contenuto di un sito, non offrono funzionalità quali il parsing, che facilita notevolmente l'analisi; inoltre non permettono l'esecuzione di codice AJAX. Con AJAX si intende una tecnica di sviluppo software che permette di realizzare applicazioni web interattive, questo è reso possibile tramite uno scambio di dati tra browser e server che permette l'aggiornamento dinamico di una pagina web, senza avere interazione con l'utente. Tipicamente realizzato in Javascript, permette di ridurre la mole di dati scambiati, in quanto è possibile inviare richieste al server per ottenere solo i dati necessari per una situazione specifica; un esempio è l'ordinamento di dati all'interno di una tabella che può essere realizzato tramite Javascript, invece che inviando una query al server che restituirebbe una nuova tabella ordinata. Il web driver che si è deciso di utilizzare è Selenium, uno degli applicativi più diffusi nel campo della navigazione automatica. Selenium è disponibile in formato open source e dispone di driver per i linguaggi di programmazione più diffusi, come Java e python. È inoltre compatibile con diversi browser (per questo progetto si è scelto di utilizzare Firefox) e facilmente configurabile. E' stato scelto di utilizzare Selenium anche perché esiste una libreria di Go (il linguaggio in cui è stato scritto il software) che permette di gestire il comportamento del suo web driver direttamente dal codice, in modo semplice e altamente personalizzabile: permette infatti di gestire variabili come il tempo di attesa per il caricamento di una pagina o di imporre delle condizioni che devono essere sodisfatte prima di procederne all'analisi; un esempio può essere aspettare il caricamento di un certo numero di elementi che appartengono a una specifica classe, come i link ad altre pagine web.

2.3 Database

Una volta estratte le informazioni dai vari siti web, è stato necessario conservarle in un database. Per la scelta del *Database Managment System* (DBMS) si è puntato su uno basato sui grafi piuttosto che relazionale. Nel caso di un database relazionale viene fornito un modello che permette di specificare in modo diretto i dati e le query. Molti adottano il linguaggio SQL (Structured Query Language) per la definizione dei dati e la scrittura delle query; nonostante sia uno standard ne esistono molte implementazioni, infatti capita spesso che particolari dettagli implementativi (alcuni sono case sensitive, altri no) oppure nel comportamento di alcune *clause*. La sintassi dell'SQL è composta da diversi elementi: clausole che sono il componente costituente delle query e descrivono l'azione da eseguire, espressioni che possono produrre valori scalari o tabelle, predicati usati per specificare condizioni logiche che condizionano il comportamento di statement e query. Infine, i due macro-componenti principali: gli *statement* che possono avere un effetto persistente sugli schemi e sui dati, controllare il flusso del programma e la sessione e le query che hanno il compito di recuperare i dati in base a specifici criteri. In questo modello, i dati vengono conservati all'interno di tabelle in cui una chiave univoca identifica ogni riga, chiamata anche *record* o *tupla*; le colonne vengono chiamate attributi. Le righe rappresentano un'istanza del tipo di entità della tabella e le colonne rappresentano i valori attribuiti a quell'istanza. Un concetto chiave di questo tipo di database sono le chiavi primarie che identificano in modo univoco e non ambiguo una riga all'interno della tabella, quindi un'istanza da tutte le altre. Può essere un attributo che naturalmente appartiene all'entità, per esempio nel caso di una persona può essere il codice fiscale che è unico per ogni individuo; nel caso un'informazione del genere sia assente è possibile ripiegare su una chiave surrogata, in questo caso si crea un nuovo attributo con valore unico per ogni entità, può essere il caso di un identificatore che viene incrementato ad ogni nuovo record. Altra variante è la chiave composta che è formata da due o più attributi che, insieme, identificano l'istanza. Altro elemento chiave sono le *foreign key* dove una componente della chiave primaria di un'entità coincide con un attributo di un'altra entità; questo sistema viene spesso utilizzato per definire la

primary key di una relazione. Strumento fondamentale che i database mettono a disposizione sono gli indici che permettono di velocizzare l'accesso ai dati. Gli indici possono essere creati come qualsiasi combinazione di attributi su una relazione; le query che filtrano su un campo indicizzato possono trovare le tuple che corrispondono tramite l'indice, senza controllarle una alla volta (in modo sequenziale); il metodo più comunemente utilizzato per implementare un indice è il B+ tree che implementa una struttura ad albero le cui foglie rappresentano i record.

Alla fine, si è scelto di utilizzare un database basato su grafo in quanto ideale per conservare questo tipo di informazioni, in quanto le relazioni di citazione tra gli articoli creano una struttura ad albero in cui gli articoli sono i nodi, le citazioni sono gli archi e i metadati sono le proprietà. Avendo una struttura così simile anche le operazioni di ricerca saranno ottimizzate; infatti molti linguaggi specifici per questi tipi di database hanno una sintassi che aiuta notevolmente il programmatore nella scrittura di query come il calcolo del percorso minimo tra due nodi o il calcolo del grado del grafo. Tra i vari dbms disponibili è stato scelto Neo4j sia per la sua popolarità (è uno die più usati al mondo), sia perché sono disponibili driver per diversi linguaggi di programmazione che permettono di inviare istruzioni direttamente dal codice. Inoltre, è semplice da usare e permette di mostrare i risultati delle query sotto forma di grafo interattivo, tramite l'interfaccia mostra il grafo risultante colorando i nodi e le relazioni tra essi.

2.3.1 Database basati su grafi

Questo tipo di database utilizza gli elementi tipici della teoria dei grafi per memorizzare le informazioni: nodi, proprietà e relazioni. Quelli che in database relazionale sarebbero i record delle tabelle, qui vengono rappresentati come entità con proprietà (campi); le relazioni tra tabelle diventano relazioni tra singole entità. La qualità più apprezzata di questo modello è l'intuitività con cui si riescono ad individuare il rapporto tra i dati, particolarmente utile tra dati fortemente interconnessi. Alcuni dei linguaggi maggiormente utilizzati sono Sparql e Cypher.

Esistono principalmente due modelli per conservare i dati:

- Labeled: ai nodi e alle relazioni vengono assegnate delle etichette così da poterli raggruppare con maggiore semplicità; è il sistema più elementare ed è quello adottato da Neo4j.
- Resource Description Framework (RDF): i dati vengono conservati sotto forma di triplette:

  nodo (soggetto) – relazione (predicato) – nodo (oggetto)

  anche qui nodi e relazioni possono avere delle proprietà. I singoli nodi possono essere identificati da stringe o URI (stringhe che identificano in modo non ambiguo una risorsa).

Sono particolarmente utili quando si ha necessità di eseguire query graph-like, come la ricerca del percorso minimo tra due nodi. Inoltre, non richiedono un grande lavoro a livello di progettazione del database, questo rende più semplice l'aggiunta di nuovi dati senza paura di perdere funzionalità.

Come nei database relazionali, per mantenere la consistenza dei dati si utilizza il modello ACID; tuttavia la sua implementazione è molto diversa. ACID garantisce al database le seguenti caratteristiche:

- Atomic: tutte le operazioni in una transazione devono avere successo o il sistema torna allo stato precedente all'esecuzione della prima operazione.
- Consistent: al termine di ogni transazione, il database deve trovarsi in uno stato coerente, quindi non devono esserci contraddizioni tra i dati archiviati (inconsistenza).
- Isolated: ogni transazione deve essere indipendente dalle altre, così il fallimento di una transazione non influisce sulle altre.
- Durable: i cambiamenti apportati da una transazione avvenuta con successo non potranno andare persi nel tempo (persistenza).

Queste proprietà garantiscono che i dati prodotti da una transazione completata con successo siano consistenti e conservati in modo permanente su disco.

2.4 Linguaggio di programmazione

Il linguaggio di programmazione più adatto alla produzione del software relativo a questo progetto deve soddisfare due caratteristiche fondamentali: deve essere integrabile con librerie/classi che permettano di interagire con i vari componenti (web scraper e dbms) e deve implementare una qualche forma di multi-theading con buone prestazioni. Sarebbe inoltre preferibile se avesse elementi tipici dei linguaggi dinamici come la reflection, ma compilato così da migliorarne le prestazioni in fase di esecuzione.

3 Scelte implementative

Ora verranno analizzati gli applicativi e le risorse che implementano gli strumenti e le funzionalità descritte in precedenza.

3.1 Motori di ricerca

La scelta dei motori di ricerca è caduta su Microsoft Academic e Google Scholar principalmente perché entrambi trattano un'ampia varietà di argomenti (prevalentemente scientifici) e sono liberamente consultabili da chiunque.

3.1.1 Google Scholar

Google Scholar è un motore di ricerca accessibile liberamente che tramite parole chiave specifiche consente di individuare vari tipi di pubblicazioni come articoli scientifici, tesi di laurea e libri. Consente inoltre di reperire articoli da una vasta gamma di case editrici che si rivolgono al mondo dello studio e della ricerca da associazioni scientifiche e professionali e università, oltre che nella galassia di articoli scientifici e culturali distribuiti nel Web.

Uno dei suoi pregi è quello di fornire, tramite una ricerca Web, l'accesso gratuito a una versione full text dell'articolo senza il permesso del giornale che lo ha pubblicato. Questo perché cerca copie degli articoli indicizzati anche su siti considerati non sicuri. Scholar non fornisce una lista dei periodici da cui prende gli articoli perché si limita a fare una lista di tutte le pubblicazioni dalla sua ricerca per il Web. Oltre alla ricerca classica in cui cerca all'interno del proprio indice le parole inserite dall'utente, permette anche una Ricerca Avanzata. Qui è possibile specificare le parole che devono comparire nel titolo e/o nel corpo, gli autori, gli editori e la data di pubblicazione. (9) Tra le informazioni dei risultati compare il numero degli articoli che hanno citato il risultato, questi articoli possono essere recuperati dal link pertinente. Altro link rilevante è quello che porta agli articoli correlati cioè quelli Scholar ritiene più "simili".

L’algoritmo che Scholar usa per il ranking non è pubblico ma se ne conosco i parametri: il numero di citazioni è quello tenuto maggiormente in considerazione; infatti articoli citati spesso compaiono con molta più frequenza tra i risultati delle ricerche. Di conseguenza Scholar sembra essere più indicato per la ricerca di articoli che seguono la corrente principale piuttosto che quelli che propongono una visione alternativa. (10) Tiene inoltre conto del testo dell’articolo, dell’autore e della rivista.

Non tutti gli articoli mostrati tra i risultati sono accessibili in quanto appartengono a riviste commerciali, in questi casi le uniche informazioni disponibili sono parte dell’abstract e le citazioni. Per accedere all’articolo è necessario pagare. Altra pecca di Scholar è che non tutti i risultati sono articoli accademici, include anche: annunci di notizie, presentazioni in PowerPoint e materiale non pubblicato. Sarebbe quindi necessario filtrare i risultati per discernere le pubblicazioni dal resto. Scholar attribuisce grande importanza al numero di citazioni quando calcola il ranking e per questo è stato accusato di rafforzare l’effetto Mathew: le pubblicazioni più citate compaiono nelle prime posizioni, mentre quelli più recenti difficilmente appaiono in cima; di conseguenza ricevono meno attenzione da parte degli utenti e quindi meno citazioni. Scholar (come molti altri motori di ricerca) tiene particolarmente ad evitare che software esterni a lui lo sfruttino per raccogliere informazioni, per questo implementa il controllo reCAPTCHA che richiede la verifica da parte dell’utente nel caso rilevi un’attività sospetta. Dal 2018 Google ha iniziato a testare un sistema invisibile di reCAPTCHA che non richiede nessuna verifica visuale, al suo posto questo nuovo sistema monitora attivamente le azioni dell’utente e gli assegna un punteggio che rappresenta la probabilità che sia un robot.

Google Scholar è particolarmente vulnerabile allo spam. Ricercatori dell’università della California, Berkley e Otto-von-Guericke hanno dimostrato che i conteggi delle citazioni sul motore di ricerca possono essere manipolati ed è possibile creare articoli privi di significato ma che vengono comunque indicizzati. È quindi possibile creare gruppi di articoli falsi che si citano a vicenda e contenenti determinate parole chiave per invalidare il contenuto dell’indice di Scholar. (11)

3.1.2 Microsoft Academic

Microsoft Academic è un motore di ricerca gratuito per articoli scientifici e letteratura accademica, sviluppato dalla Microsoft Research. La raccolta dei documenti avviene tramite strumenti che sfruttano l'intelligenza artificiale per processare i documenti scoperti da Bing (crawling), riconoscere quelli accademici e aggiungerli alla sua base di conoscenza. Le informazioni che riesce a ricavare dalle pubblicazioni sono molteplici: gli URL dei sorgenti differenziandoli tra risorse presenti come pagine Web e scaricabili come PDF, l'abstract, la data, gli autori, il giornale e le citazioni.

È possibile effettuare diverse operazioni sui risultati della ricerca: ordinarli dal più recente, per rilevanza e per numero di citazioni. Academic fornisce ulteriori informazioni oltre ai risultati in sé: sulla sinistra mostra autori, affiliazioni, riviste e altre entità presenti nel suo database correlate agli argomenti della ricerca; mentre sulla destra mostra eventuali informazioni sull'autore (nel caso una delle parole facesse match con un autore).

Queste informazioni sono conservate nel Microsoft Academic Graph (MAG), un grafo contenente più di 210 milioni di pubblicazioni e 256 milioni di autori. L'essere costruito in modo automatico comporta degli svantaggi come perdersi delle citazioni (solo 60 milioni di articoli hanno delle citazioni) e la mancanza di disponibilità delle informazioni delle affiliazioni. D'altra parte, vanta una vasta base di pubblicazioni ricca di metadati sugli articoli (al contrario di Google Scholar) che copre bene diversi campi di ricerca, in particolare per le discipline scientifiche. Anche la disambiguazione delle affiliazioni e degli autori ha un buon livello di correttezza.

Al contrario di Google Scholar, Academic memorizza anche i campi di studio (o i topic) di un articolo. Stando alla definizione data da Microsoft:" Topics are organized in a non-mutually exclusive hierarchy with 19 top-level fields of study" (12). In pratica i campi sono organizzati in una gerarchia divisa su 4 livelli: in cima risiedono pochi (19) campi più generici mentre scendendo nei livelli inferiori si ha via via una maggiore granularità. I topic più diffusi (ordinati per numero di occorrenze) sono: medicina, biologia, scienza dei materiali, chimica, ingegneria,

computer science. È importante notare che il dataset viene costantemente aggiornato e con esso la classifica, anche se sono sempre gli argomenti a carattere scientifico a dominare le prime posizioni. I nuovi topic vengono scoperti da sistemi che sfruttano l'intelligenza artificiale per effettuare una comprensione semantica del contenuto delle pubblicazioni; questa analisi ha anche lo scopo di aggiornare la gerarchia in modo corretto, disponendo i topic secondo un ordine logico di specificità (es. biologia ⟶ botanica ⟶ coltivazione)

Nel Microsoft Academic Graph sono presenti più di 1,9 miliardi di coppie di citazioni e in media ogni entità (articoli, libri, conferenze, …) viene citata 9,33 volte. Tuttavia, una parte significativa del grafo è disconnessa dal resto: sono presenti molti articoli che non citano, né vengono citati; è infatti comune trovare articoli privi di citazioni anche quando si cercano argomenti molto popolari. Con i vari aggiornamenti del grafo, il numero delle citazioni è aumentato notevolmente; più del numero delle pubblicazioni. Se da un lato questo suggerisce un costante miglioramento dei contenuti, in realtà può anche indicare una raccolta dei dati non meticolosa. (13)

Una recente feature offerta da Microsoft Academic permette di ottenere diverse informazioni statistiche e classifiche. Nella sezione "preview" del sito, è disponibile una pagina dedicata alle pubblicazioni in cui è possibile specificare un campo di studio e ricevere delle informazioni al riguardo. Tra queste ci sono un istogramma con il numero di pubblicazioni divise per anno e diverse top 10: migliori autori, riviste, istituti e conferenze in cui è stato trattato quell'argomento. Purtroppo, non è possibile specificare direttamente il topic ma, partendo da quelli top-level, è necessario cliccare su quello che sembra più legato a quello interessato. In questo modo ne vengono proposti nuovi di più specifici o correlati a quello precedente. Da un lato questo tipo di ricerca aiuta l'utente inesperto o che non sa di preciso cosa cercare (o ne ha solo una vaga idea), dall'altro permette di scoprire nuovi argomenti legati a quello che si sta cercando. (14)

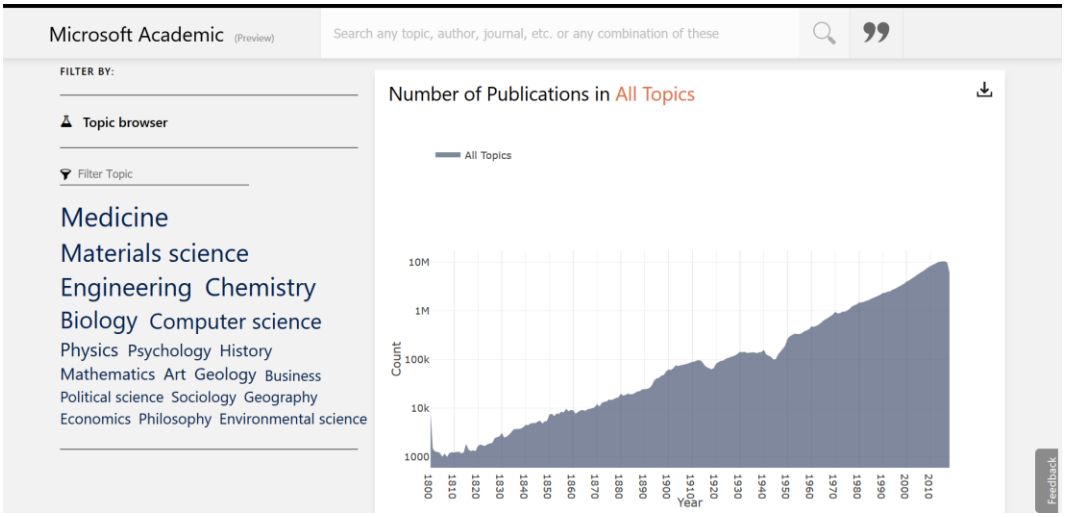


FIG. 4: Aumento del numero di pubblicazioni indicizzate da Microsoft Academic

3.2 Web scraping

Come strumento per effettuare questa operazione è stato scelto Selenium, non solo per la sua popolarità, ma anche perché è facilmente utilizzabile tramite la maggior parte dei linguaggi di programmazione e si interfaccia con i browser più diffusi.

3.2.1 Selenium

Selenium è uno dei software per l'automazione di browser più diffusi al mondo. La sua principale funzione è svolgere dei "test", permette quindi di inviare al browser una serie di operazioni da fargli eseguire in modo automatico. Questo si rivela molto utile quando si vogliono testare le funzionalità del proprio sito web in fase di sviluppo: come l'applicazione web reagisce a diversi tipi di input, misurare i tempi di risposta e così via. (15)

Il software è disponibile per tutte le maggiori piattaforme: Winows, Linux e macOS ed è disponibile in formato open-source con licenza Apache 2.0.

Selenium è un insieme di diversi componenti:

- Selenium IDE: è un plugin disponibile per Firefox e Chrome e offre un'interfaccia grafica che aiuta lo sviluppatore a realizzare i test. Tra le

principali funzionalità che offre ci sono: strumenti per la registrazione delle sessioni e per la scrittura/debugging dei test. Solo a partire dal 2018, Selenium IDE iniziò ad essere mantenuto in modo attivo dalla comunità.

- Selenese: è il linguaggio in cui sono scritti i test, permette di inviare comandi al browser e esaminare il codice delle pagine. Sono disponibili svariate API per utilizzare Selenium senza utilizzare necessariamente Selense; per i maggiori linguaggi esistono delle versioni ufficiali aggiornate periodicamente (come per il C++, Java e python). Esistono però API non ufficiali per i linguaggi meno diffusi.
- Selenium Webdriver: è il cuore dell'applicazione, accetta comandi (inviati tramite Selenese o API) e li invia al browser. Per poter interagire con i vari browser ha bisogno di driver specifici (es. per Firefox e necessario geckodriver), questi gli permettono di tradurre le sue istruzioni generiche in una sintassi specifica per quel browser. Dalla versione 2.0, si è deciso di utilizzare funzionalità al livello del sistema operativo per interagire col browser, invece che utilizzare dei comandi in JavaScript; questo ha ridotto notevolmente il numero delle funzioni disponibili ma a favorito la creazione di versioni di Selenium ad och per svolgere compiti specifici.
- Selenium Remote Control (RC): è un server che acetta comandi via http. Scritto in Java, permette di scrivere test automatizzati per un'applicazione web in qualsiasi linguaggio di programmazione.

Per lo sviluppo di questo progetto è stato sufficiente utilizzare il WebDriver, rinominato dai suoi sviluppatori Selenium Standalone Server, che dalla versione 3.0 in poi non necessita più del Remote Control, ma è in grado svolgere tutte le operazioni in modo autonomo. Sono state utilizzate diverse versioni nel corso del tirocinio ma quella definitiva è la 3.141.59 e consiste di una singola applicazione java. (16)

3.2.2 WebDriver

Il webdriver è un'interfaccia di controllo remota che permette di interagire con lo user agent; fornisce una piattaforma e un linguaggio che permettono ai programmi

di inviare istruzioni al browser web. Permette di interfacciarsi e modificare gli elementi della pagina web (DOM), consente così di eseguire test automatizzati e script. (17)

Implementa un protocollo che permette la comunicazione tra:

- Local end: lato client, gestito tramite librerie che implementano le API.
- Remote end: server side, si dividono in nodi intermedi (che agiscono come proxy) e endpoint (destinazione).

Tra le principali funzionalità ci sono:

- Non essendo legato ad uno specifico framework, può essere facilmente integrato con framework per il testing come JUnit e TestNG.
- È in grado di gestire gli alert di Javascript, più frame e più finestre contemporaneamente.
- Permette di trovare le coordinate degli elementi nella pagina.
- Simulare il comportamento del mouse e la pressione dei tasti della keyboard.
- Supporta il testing per l'ambiente Android e iOS, il drag-and-drop, gli elementi AJAX e la navigazione tra le pagine.

Per il corretto funzionamento di un web driver sono anche necessari dei driver che gli permettano di interagire col browser e una libreria che permetta di gestirne il comportamento direttamente dal codice.

3.2.3 Geckodriver

Questo programma si comporta come un proxy: fornisce le API http descritte dallo standard W3C del protocollo dei WebDriver; il suo compito è tradurre queste chiamate in altre comprensibili a Firefox.

È possibile specificare la posizione dell'eseguibile di Firefox che si vuole usare, eventuali parametri che gli si vogliono passare e la verbosità dei file di log. (18)

3.2.4 Driver Go

Questo driver fornisce un client WebDriver per programmi scritti in Go; è disponibile su Github ed è mantenuto da "Miki Tebeka". (19)

Questo package fornisce funzioni che permettono di avviare il server di Selenium (supporta sia Firefox che Chrome) e di inviargli dei comandi. Sarà quindi possibile navigare all'interno dei siti web, interagire con le pagine e gestire i cookie. Offre anche la possibilità di ridimensionare la finestra del browser e impostare un tempo massimo di attesa per il caricamento di una pagina.

3.3 Database

Per la gestione del database si è scelto di utilizzare Neo4j perché è provvisto di un proprio linguaggio di formulazione per le query: Cypher, uno dei migliori linguaggi per l'esplorazione dei database a grafo che però DGraph non supporta ancora. I suoi punti di forza sono proprio la facile interpretabilità delle query e l'intuitività di come vengono mostrati i risultati. DGraph sarebbe stato preferibile con una grande mole di dati distribuita su più server ma ai fini del progetto è sufficiente conservare tutte le informazioni su un singolo server, data la loro piccola quantità e la disponibilità di hardware.

3.3.1 DGraph

DGraph è nato nel 2015 da uno sviluppatore della Google, con l'obbiettivo di fornire un database basato su grafo distribuito e open source. Il codice è interamente realizzato in Go ed è disponibile su Github. (20) Alcuni dei suoi obbiettivi sono l'alta scalabilità, bassa latenza con particolare enfasi sulla concorrenza. Anche DGraph fornisce le proprietà ACID, così da rendere le transazioni affidabili e persistenti; è inoltre provvisto di un'architettura distribuita, è infatti possibile distribuire i dati tra più sever e aggiungere all'occorrenza l'hardware necessario per gestire le richieste. DGraph implementa alcune funzionalità interessanti: è in grado di ridurre il numero di chiamate ai server all'interno di un cluster e favorisce l'alta

concorrenza dell'esecuzione delle query, questo gli conferisce un alto query throughput. Consente inoltre la replicazione consistente dei vari frammenti per avere più resistenza ai crash e permette l'accesso ai dati anche quando un server non è disponibile. Per interrogare il database, DGraph mette a disposizione un linguaggio che deriva da GraphQL, un linguaggio per query sviluppato da Facebook, il GraphQL+. Questo linguaggio punta a restituire le informazioni in una struttura ordinata facilmente accessibile e manipolabile da parte dell'utente, struttura che contiene tutte e sole le informazioni sulle entità che stava cercando. Purtroppo, la sintassi del linguaggio non è semplice e intuitiva come quella di altri linguaggi, quali il Cypher; la sua struttura ricorda quella di una chiamata a funzione e la struttura si complica notevolmente nel caso di query complesse. Inoltre, i risultati vengono mostrati con una struttura a grafo ma pur sempre testuale e quindi meno intuitiva di quella che producono altri dbms come Neo4j.

3.3.2 Neo4j

Neo4j è un database managment system basato sui grafi sviluppato da Neo4 Technology e stando alla classifica stilata da DB-Engines è il database su grafi più popolare al mondo. (21)

Il software è completamente scritto in java ed è disponibile open-source la versione "community" tramite licenza GPL3; fornisce backup online e una vasta gamma di estensioni, sia gratuite che commerciali. La prima release è avvenuta nel 2010, mentre la stable attuale (3.0) è stata rilasciata nel 2016.

Una volta lanciato il programma, è possibile accedere alle funzionalità del dbms tramite un qualsiasi browser web collegandosi a localhost sulla porta del protocollo scelto: http (7474), https (7473) oppure bolt (7687). Tramite l'interfaccia è possibile esplorare il database tramite query scritte in Cypher, un linguaggio che come lo Sparql, è stato pensato per database non relazionali. Neo4j è in grado di gestire un solo database alla volta, non è pensato per l'ambito distribuito in quanto allo scalare dei dati si ha uno scalamento verticale dell'applicazione server, quindi un aumento esponenziale dell'utilizzo delle risorse computazionali.

Nonostante la semplicità della sua interfaccia, si rivela intuitivo funzionale. La entry in cui è possibile scrivere la query è provvista di completamento automatico, inoltre mostra dei warning quando si cerca di eseguire query troppo onerose dal punto di vista computazionale; per esempio quando, nella selezione dei parametri (nodi e relazioni) su cui si andrà a lavorare, si effettua un prodotto vettoriale, ritrovandosi con una lista molto lunga di parametri da controllare. I risultati vengono mostrati in diversi formati:

1) Testuale: esattamente come in sql i risultati sono inseriti in una tabella che ha come nome delle colonne, quelli delle variabili in output definite nella query; mentre i risultati sono le entry.
2) Json: questo formato è particolarmente utile quando si vogliono esportare i risultati per passarli ad un'altra applicazione in modo che gli elabori.
3) GUI: il formato più intuitivo e utile per farsi un'idea di come sono distribuiti i nodi, quali relazioni li legano, quali hanno un grado maggiore, e così via; un limite di questa visualizzazione è il numero massimo dei nodi visualizzabili che è molto limitato (circa 300), se si imposta una soglia superiore sarà necessaria una quantità non indifferente di memoria RAM per evitare cali di prestazioni. Una caratteristica utile è la possibilità di colorare i nodi e le relazioni in base alle label assegnate; questo permette di evidenziare percorsi e entità.

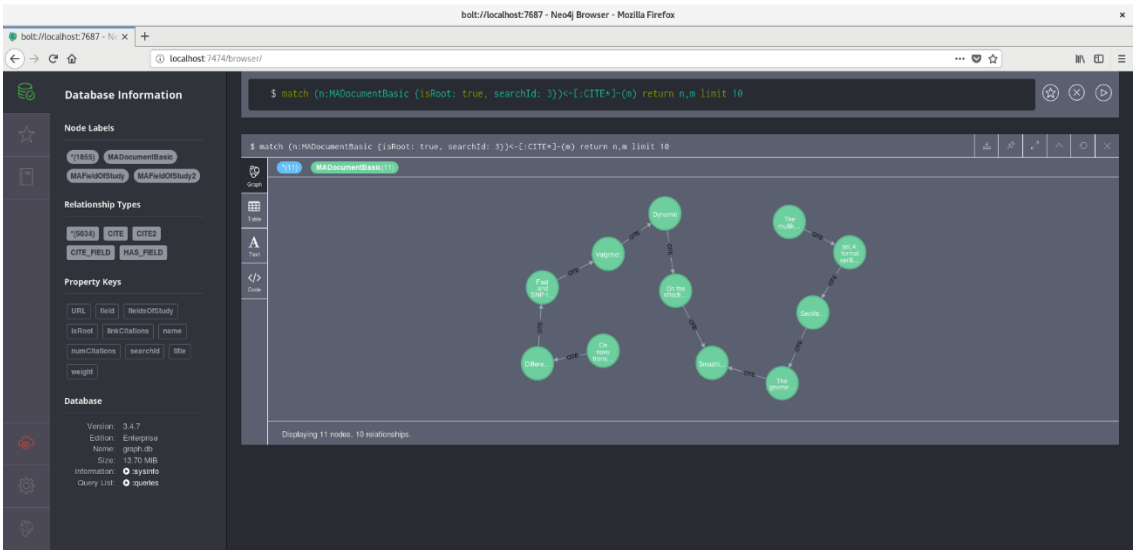


FIG. 5: Interfaccia grafica di Neo4j con grafo

Consente anche di definire degli schemi per le entità e le relazioni che permettono poi di creare indici e vincoli che migliorano le prestazioni durante la ricerca. I principali modi per esportare i risultati sono in formato json, se si vogliono passare come input per un'analisi più approfondita e png, se è necessaria solo l'immagine del grafo in output. Esiste inoltre un plugin che permette di esportare nel formato GraphML che può essere letto da applicazioni come Gephi e yEd. (22)

Le unità fondamentali di questo modello sono:

- Nodi: rappresentano le entità e possono possedere delle proprietà, può anche essere assegnata loro un'etichetta (label).
- Relazioni: servono per indicare un qualche tipo di connessione tra due nodi, anche loro possono avere proprietà e etichette. È possibile assegnarle una direzione, se si ha bisogno di un grafo diretto; ma è importante notare che è possibile avere due relazioni dello stesso tipo (stessa label) una diretta, l'altra no. Una relazione può essere attraversata in entrambi i versi, è quindi possibile specificare prima la destinazione e poi la sorgente o anche solo una delle due.
- Proprietà: sono coppie chiave valore dove la chiave è una stringa (il nome della proprietà), mentre il valore può essere di diversi tipi come string e int. Sono supportati anche gli array di tipi primitivi.
- Label: serve per raggruppare i nodi/relazioni: gli elementi con la stessa label appartengono allo stesso sottoinsieme; questo rende le interrogazioni più facili da scrivere. Le label possono essere modificate a runtime, questo permette di segnare temporaneamente alcuni elementi, magari per indicarne lo stato.
- Path: il percorso indica il sotto grafo che separa il nodo sorgente da quella destinazione e la sua lunghezza è data dal numero di salti necessari per raggiungere la destinazione. Il percorso più corto possibile è costituito da un singolo nodo privo di relazioni e ha lunghezza 0.

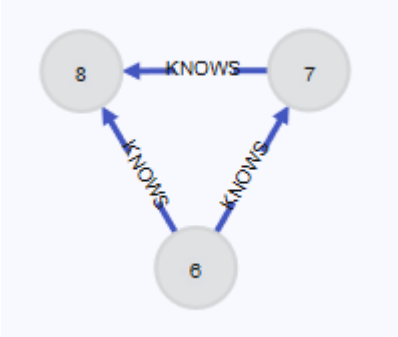
3.3.3 Cypher

Cypher è un linguaggio dichiarativo creato per interrogare database basati su grafo. È il linguaggio nativo di neo4j e punta sull'espressività e sull'efficienza nell'esecuzione delle query. Quello che l'utente fa è chiedere al database di trovargli "tutti gli oggetti simili o che gli assomigliano" a un dato pattern.

La sintassi di Cypher permette di descrivere con caratteri ASCII il pattern del sotto grafo che ci interessa esaminare, per esempio:

(a)-[:KNOWS]->(b)-[:KNOWS]->(c) , (a)-[:KNOWS]->(c)

Questa sintassi permette di individuare una struttura di questo tipo:



In Cypher è anche possibile dei punti da cui iniziare la ricerca, invece che controllare tutti i nodi del grafo per controllare se corrispondono al pattern. Questi nodi sono chiamati punti di ancoraggio e possono essere definiti tramite la clausola START. Tuttavia, dalla versione 3.2 questa possibilità è stata rimossa; al suo posto è possibile sfruttare gli indici per migliorare le normali prestazioni che si avrebbero con MATCH. Questo è possibile grazie alle ottimizzazioni della nuova versione del compilatore che permette 3 ambienti di runtime (interpreted, slotted e compiled) che sfruttano una sempre migliore gestione delle singole operazioni al fine di migliorare le performance e l'utilizzo della memoria. La modalità più performante (compiled) permette di raggruppare in modo intelligente le operazioni così da creare un nuovo piano di esecuzione che migliora notevolmente le prestazioni e il consumo di risorse; tuttavia è ancora in fase di sviluppo e non supporta tutti gli operatori e le query. Da notare che gli ultimi due ambienti sono disponibili solo per la versione Enterprise.

Come in sql, le query sono composte da clausole, alcune delle principali sono:

- MATCH serve per specificare uno o più pattern all'interno del grafo, permette di definire su quali dati lavorare.
- WHERE aggiunge dei vincoli per filtrare i risultati, ad esempio specificando il valore di certe proprietà.
- RETURN cosa restituire.

Esistono anche altre clausole che permetto di creare nuovi nodi e relazioni (CREATE) e di aggiungerli se non sono già presenti (MERGE). Sono anche presenti le clausole classiche del sql come LIMIT per limitare il numero dei risultati restituiti e ORDER BY che permette di ordinare i risultati secondo una qualche proprietà.

Le label sono anche utilizzate per la creazione di indici, che velocizzano di molto la ricerca, e la definizione dei vincoli. È importante notare che, in assenza di vincoli, le singole entità non sono identificate dal valore di una loro proprietà, come poteva essere per i database relazionali. Tipicamente esiste un identificatore comune a tutte le entità e relazioni (indipendente dalla label) che viene incrementato a ogni creazione; è infatti possibile creare molteplici entità con gli stessi valori. In questo linguaggio "null" è usato per rappresentare un valore mancate o non definito.

3.3.4 Driver Go

Questo driver è disponibile si Github ed è stato sviluppato da "johnnadratowski". Questo driver, scritto in Go, realizza un'implementazione del protocollo Bolt a basso livello; questo permette di inviare comandi a Neo4j e ricevere i risultati in risposta. Tra le caratteristiche principali si ha il pipelining dei messaggi, per migliorare la concorrenza e la possibilità di creare una pool di connessioni, così da svolgere compiti diversi contemporaneamente.

Di default le procedure si aspettano che le query in input siano scritte in Cypher, ma supporta anche le funzionalità richieste dall'interfaccia di Go "database/sql/driver". Tuttavia, è caldamente consigliato l'utilizzo di Cypher in quanto fornisce maggiori funzionalità e prestazioni. (23)

Recentemente l'autore ha comunicato che non manterrà più il codice; l'ultima versione disponibile è la 3.1.0-M02. (24)

3.4 Linguaggio di programmazione

Si è quindi scelto di utilizzare Go per la presenza di librerie per la gestione degli strumenti di ricerca, per la sua crescente popolarità, per l'ampia disponibilità di package open source e per la possibilità di creare codice concorrente in modo semplice ma non sofisticato.

3.4.1 Go

Per lo sviluppo del codice si è scelto di utilizzare il linguaggio Go, sviluppato dalla Google. (25) Disponibile dal 2012, si tratta di un linguaggio relativamente giovane le cui caratteristiche principali sono la tipizzazione statica e la compilazione; nonostante la tipizzazione statica, sono implementate procedure di salvaguardia della memoria e di *garbage collection,* una tecnica che permette di liberare zone della memoria a cui non è più possibile accedere, a causa per esempio della perdita del riferimento alla zona stessa e che ha l'obbiettivo di ottimizzare l'utilizzo della memoria a scapito di maggiore costo computazionale. Questo linguaggio adotta elementi di sintassi tipici dei linguaggi dinamici come la *type inference* che permette la dichiarazione e inizializzazione di variabili in modo più conciso rispetto ad altri linguaggi, infatti non è sempre necessario dedurre il tipo della nuova variabile ma può essere dedotto dalla variabile/valore a destra dell'assegnamento. Altre caratteristiche sono una compilazione veloce e una gestione semplice dei pacchetti da remoto tramite comando da terminale (go get), comando che fornisce anche accesso alla documentazione online. Ha una sintassi molto rigida e punta a minimizzare la quantità di codice scritta dal programmatore; a tal scopo non permette nemmeno di compilare se individua variabili dichiarate ma mai utilizzate o librerie importate di cui non si utilizzano neanche una funzione. Queste ultime caratteristiche non lo rendono adatto a una scrittura di codice veloce ma a una più lenta che mira all'ottimizzazione e alla chiarezza; infatti Go mette a disposizione un secondo comando (go fmt) che permette di modificare la formattazione del

codice sorgente per adattarla agli standard di Go, come ad esempio non permette di tenere le parentesi graffe aperte da sole su una linea. Go dispone dei tipi principali presenti nei linguaggi di programmazione come interi (in diverse dimensioni), float, booleani e stringhe. È possibile definire sia array statici, che non potranno essere modificati in futuro e array dinamici o *slice* che consistono sostanzialmente in una struttura che punta a una porzione di array statico. Sono presenti anche tipi più complessi come i puntatori, le hash table (*map*) e i canali che consentono di scambiare dei valori tra processi di Go concorrenti, sono sostanzialmente dei buffer con dimensione finita in cui è possibile inserire/estrarre elementi. Go presenta anche elementi classici della programmazione ad oggetti come l'ereditarietà e il polimorfismo ma non ha il concetto di classe, almeno nel senso tradizionale del termine. Sono presenti le strutture (gruppi di variabili con un nome) a cui è possibile assegnare delle funzioni: durante la loro definizione se si specifica un tipo di variabile (non per forza una struttura) si assegna quella funzione a quel tipo e ogni variabile potrà invocarla. Il polimorfismo è implementato tramite le interfacce che sono oggetti equivalenti alle strutture ma contengono una serie di dichiarazione di metodi. Qualunque tipo/struttura che implemeta (a cui sono stati asseganti) tutti i metodi dichiarati dentro un'interfaccia, avrà come tipo dinamico quello dell'interfaccia e sarà quindi possibile assegnare a variabili del tipo dell'interfaccia quella variabile. Da notare che non è necessario, né richiesto dire in qualche modo che un tipo implementa un'interfaccia; questo lavoro viene svolto a tempo di compilazione e permette di avere polimorfismo a runtime. Altra caratteristica fondamentale è la presenza di librerie che permettono di interagire con i vari componenti (Neo4j e Selenium) direttamente dal codice. (26) Data la filosofia del linguaggio, queste librerie sono disponibili in formato open source su Github; in modo che chiunque possa contribuire al loro sviluppo e mantenimento. Purtroppo, è necessario notare che lo sviluppatore proprietario del repository dei driver per Neo4j ha annunciato che non potrà più continuarne lo sviluppo. Il software è attualmente funzionante ma è possibile sorgano problemi con versioni successive di Neo4j.

3.4.2 Concorrenza in Go

Go permette l'esecuzione concorrente di più funzioni all'interno dello stesso processo, queste vengono chiamate *goroutine* quando una funzione viene invocata con il prefisso go davanti (es. go add(n1, n2)), l'esecuzione della funzione viene delegata da Go a un thread. Queste routine hanno dei vantaggi rispetto ai thread: occupano minore spazio all'interno dello stack e vengono assegnate dinamicamente ai thread del sistema tramite un sistema di multiplexing, così quando una rimane in attesa può essere sostituita da un'altra pronta per l'esecuzione. La comunicazione, e di riflesso la sincronizzazione, tra queste routine è resa possibile proprio dai canali che condividono, possono essere utilizzati per scambiarsi informazioni sui risultati raggiunti fino a quel momento o più semplicemente possono costringere una routine ad aspettarne un'altra. Infatti, la lettura (x <- channel) da un canale vuoto fermerà l'esecuzione di una routine finché in quel canale non ci sarà almeno un elemento da poter leggere; stesso risultato si avrà per la scrittura (x -> channel) in un canale pieno, la routine resterà in attesa finché non si libererà una posizione. Sono presenti anche altre librerie che forniscono mezzi per la gestione della concorrenza, alcuni degli strumenti forniti sono mutex e lock.

4 Dettagli implementativi

In questa sezione mostrerò come è stato diviso il software (*package*) e quali funzionalità offre ciascun componente. I package che compongono l'applicazione sono divisi in base agli strumenti che gestiscono: il primo definisce la struttura dei dati, in particolare quali metadati gestisco per le varie tipologie di articoli; il secondo gestisce l'interazione col database; il terzo comprende tutte le funzioni che interagiscono col web driver e l'ultimo comprende le funzionalità principali.

4.1 Struttura Dati

In questo modulo vengono definite le strutture dei dati relative alle due tipologie di articoli che si andranno a trattare: quelli provenienti da Google Scholar e quelli di Microsoft Academic.

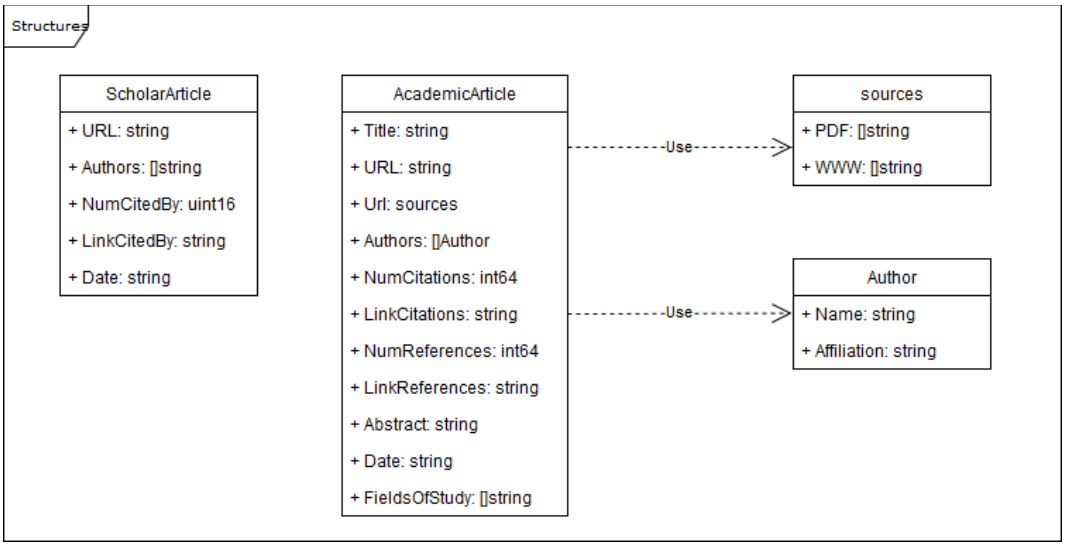


FIG. 6: Struttura dati relativa alle pubblicazioni

Le pubblicazioni relative al motore di ricerca di Google sono provviste di meno metadati rispetto a quelle di Microsoft. Le informazioni che si riescono ad estrarre sono: l'URL della pagina che permette di accedere al sorgente della pubblicazione, alcuni nomi degli autori dell'articolo (con nome si intende la stringa che Scholar utilizza per identificare l'autore, che tipicamente consiste di "nome cognome"), il

numero di articoli che lo citano e l'URL della pagina di Scholar permette di accedere alla lista degli articoli che lo citano.

Per gli articoli provenienti da Academic Microsoft si è pensato a una struttura più complessa che permettesse di gestire al meglio quante più informazioni possibili. I metadati estratti sono: il titolo dell'articolo, l'URL della pagina di Academic che contiene ulteriori informazioni su di esso, una lista di tutti i sorgenti dell'articolo che il motore di ricerca è riuscito a trovare (divisi in risorse PDF e Web), la lista degli autori che oltre a contenerne il nome, indica anche l'ente nel quale è stato svolto il lavoro di ricerca, numero e indirizzo della pagina di Academic degli articoli che lo citano (come per Scholar), numero e indirizzo della pagina di Academic degli articoli che cita (non presente su Scholar), l'abstract e i vari campi di studio associati a esso.

4.2 Gestione Database

4.2.1 Struttura dati

All'interno del database sono presenti tre tipologie di nodi (ognuno identificato dalla propria label): quelli relativi agli articoli di Scholar, quelli di Academic e i campi di studio. Gli articoli di Scholar sono collegati tra loro mediante un arco diretto che rappresenta la relazione di citazione; quelli di Academic possono avere due tipologie di arco diretto: la prima rappresenta sempre una relazione di citazione, la seconda invece li collega ai relativi campi di studio (nodi). Nelle proprietà dei nodi saranno inseriti i metadati relativi a ogni articolo, fatta eccezione per gli autori che saranno dei nodi legati agli articoli pubblicati da degli archi appositi.

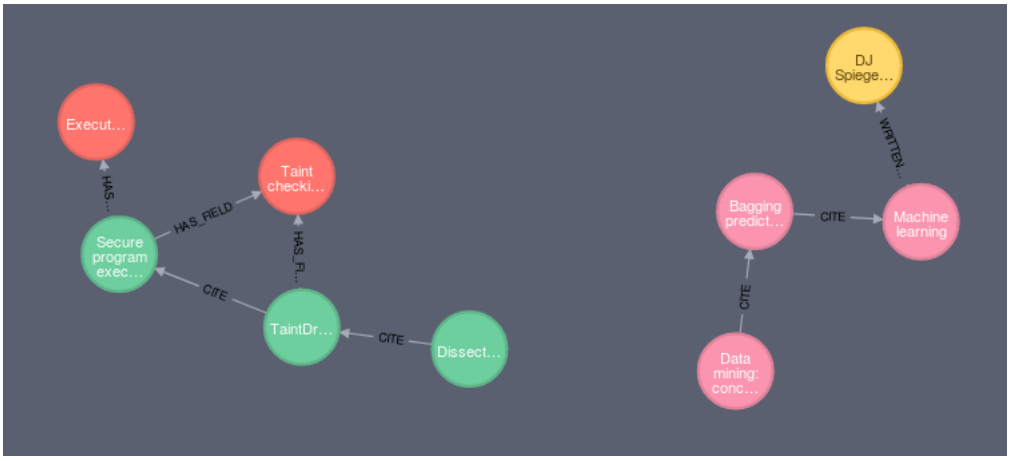


FIG. 7: Rappresentazione dei dati nel database

Nella figura prima riportata vengono mostrati i principali componenti memorizzati nel database: in verde gli articoli di Academic mentre in rosa quelli di Scholar, in rosso i campi di studio e in giallo gli autori. Sono anche presenti le relazioni tra articoli, tra articoli e autori, tra articoli e topic.

4.2.2 Funzionalità

Questo modulo fornisce le funzionalità necessarie per gestire, tramite query scritte in linguaggio Cypher, il flusso di dati diretto e proveniente dal database. Alcune delle sue funzioni di base permettono di: aprire una connessione verso il database e successivamente di chiuderla in sicurezza, permette di cancellare tutti i dati presenti o solo quelli relativi a una specifica ricerca dove per ricerca si intende tutto il processo che, partendo da un documento iniziale, porta ad esplorare quelli che lo citano e sceglierne alcuni in base a uno specifico criterio; per ciascuno di essi si ripeterà il processo di esplorazione fino al raggiungimento di una qualche soglia. È presente anche la possibilità di aprire una pool di connessioni verso il database così da poter realizzare la concorrenza.

Una delle funzionalità principali permette di ricavare/stampare a video la classifica dei campi ordinata per numero di occorrenze; è possibile stampare la classifica per intero (scelta sconsigliata dal momento che in un singolo grafo di ricerca possono comparire anche 300 *topic*) o solo una top. Il conteggio del numero di occorrenze

di un topic è dato dal numero di archi che lo collegano ogni articolo distinto, per questo motivo è possibile calcolarla solo per una singola ricerca: perché lo stesso articolo (e quindi le sue relazioni) si possono ripetere nell'arco di più esplorazioni. È anche possibile ottenere la classifica degli autori ordinata per numero di pubblicazioni a loro collegate, è disponibile sia in formato completo (tutti gli autori) o solo una top. Questa funzionalità permette quindi di avere un'idea degli autori più produttivi o celebri per un dato argomento, ma permette anche all'utente di scoprire autori di cui prima ignorava l'esistenza.

```
What research do you mean?      [ 1 - 5 ]
5
Insert the number of fields to show (-1 to see all fields):
10
RANKING:
SCORE   FIELD
194     Computer science
130     Computer security
67      Real-time computing
54      Internet privacy
46      Android (operating system)
34      Distributed computing
33      Exploit
32      Malware
32      Software
31      Cloud computing
```

FIG. 8: Classifica dei 10 campi più popolari

L'aggiunta di una pubblicazione al database avviene in modo diverso in base al motore di ricerca da cui proviene, dal momento che i metadati raccolti sono diversi. Diverse saranno anche le relazioni che si andranno a creare tra i nodi: in entrambi i casi sarà presente un arco che rappresenta la citazione ma solo per Academic si avranno i campi di studio. In entrambi casi verranno aggiunti gli autori come nodi a sé stanti e non come proprietà; nodi che saranno collegati agli articoli tramite apposite relazioni.

4.3 Web Driver

Questo modulo fornisce principalmente due funzionalità: avviare il web driver in modo corretto e raccogliere i metadati dai siti web. Nella fase di avvio, oltre al browser (Firefox) e ai driver necessari (geckodriver), sono stati impostati dei valori di default di attesa per il caricamento di una pagina; questi valori indicano quanto

tempo aspettare al massimo prima che tutti gli elementi della pagina si siano caricati. Nel prossimo capitolo verranno discussi più nel dettaglio.

La raccolta delle informazioni su una singola pubblicazione avverrà in modo diverso, in quanto cambiano i metadati presenti sul search engine; in particolare su Academic questa collezione sarà molto più lenta in quanto le informazioni sono contenute nelle singole pagine degli articoli che è quindi necessario visitare singolarmente. Invece su Scholar tutti i metadati sono già presenti nella pagina dei risultati. Sono disponibili diversi modelli di comportamento per la raccolta dei documenti citanti, di base tutti iniziano dalla pagina del motore di ricerca che contiene questi documenti, ciò che li differenzia è la condizione di stop. Condizione che dipende dalle funzionalità descritte nel capitolo 3.4 ma che si possono riassumere in: raggiungere un numero massimo di documenti collezionabili oppure tramite due tipi di soglie. Entrambe le soglie lavorano sul numero di citazioni: la più semplice imposta un threshold minimo così che la ricerca si ferma quando incontro un articolo che scende sotto la soglia; nel secondo caso viene preso il numero di citazioni massimo tra gli articoli recenti e si imposta una soglia minima basta su una percentuale di questo valore (sono presenti delle osservazioni su questi parametri nel capitolo successivo). In entrambi i casi i documenti vengono ordinati per numero di citazioni decrescente così da ottimizzare la ricerca.

Ultima funzionalità di rilievo è la ricerca del documento da cui far partire la ricerca; che può avvenire in due modi: l'utente può decidere il documento passando l'indirizzo della sua pagina (Microsoft) o l'indirizzo della pagina di articoli che citano quello che gli interessa (Google, in questo caso le informazioni sull'articolo iniziale dovranno essere inserite manualmente). In alternativa è possibile impostare delle parole chiave che verranno cercate sul motore di ricerca specificato, il primo articolo a comparire tra i risultati sarà quello iniziale.

4.4 Funzionalità Principali

4.4.1 Produzione del grafo di ricerca

Il grafo di ricerca è l'insieme delle pubblicazioni, con le relative relazioni di citazione, che vengono raccolte durante lo svolgimento di una singola ricerca. La ricerca può iniziare in due modi diversi. Nel primo caso l'utente inserisce manualmente le informazioni sul documento (può anche inserire solo alcuni dei metadati che gli interessano) ma dovrà obbligatoriamente inserire: per Scholar l'indirizzo della pagina che contiene gli articoli che lo citano, per Academic l'indirizzo della pagina dell'articolo. Nel secondo caso, verranno cercate delle parole chiave nella pagina principale del motore di ricerca e verrà preso il primo risultato come pubblicazione iniziale (questa funzionalità è stata utilizzata solo per fare dei test). La ricerca si può svolgere in modi diversi, ma quello principale consiste in una ricerca in profondità iterativa delle citazioni. Per ogni articolo sono stati esplorati quelli che lo citavano, di questi sono stati presi solo quelli che soddisfacevano certe condizioni. Il parametro principale di cui si è tenuto conto è stato il numero di citazioni di un articolo, per questo è stato necessario ordinare i risultati secondo questa caratteristica prima di procedere all'analisi della pagina. Una volta raccolti, gli articoli citanti sono stati aggiunti al database come nodi e per ognuno di essi è stata creata una relazione di citazione (arco). I metadati di ogni articolo vengono memorizzati come proprietà all'interno dei nodi; mentre per gli autori vengono creati nodi a parte a cui viene aggiunta una relazione con l'articolo scritto. Solo per Microsoft Academic si creano dei nodi per i campi di studio (topic) a cui viene aggiunta una relazione per ogni articolo che è attinente a quel campo.

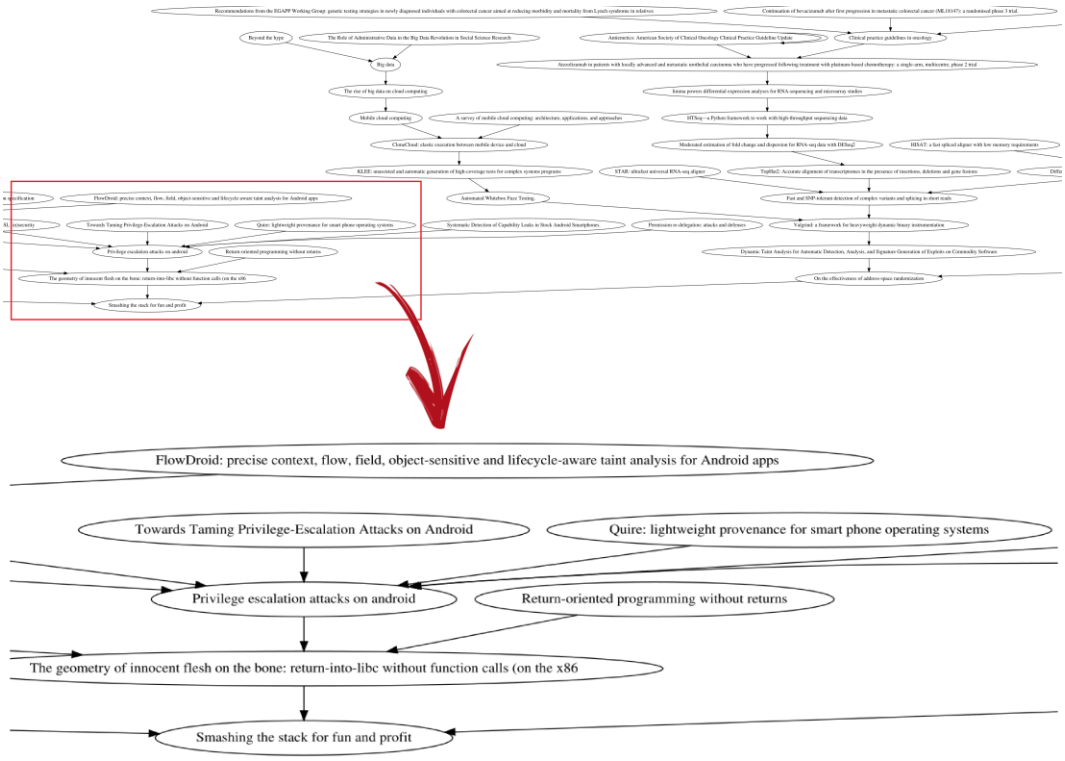


FIG. 9: Porzione di un grafo di ricerca

Sono stati realizzati sostanzialmente sue metodi di ricerca secondari. Il primo, e anche il più rozzo, consiste nell'esplorare gli articoli seguendo le citazioni, come in precedenza, ma raccoglie sempre lo stesso numero di articoli citanti: per ogni pubblicazione colleziona i primi n articoli che incontra e ripete il processo per ognuno di essi, fino a raggiungere una certa quantità di articoli. Il secondo colleziona articoli fino a raggiungere una certa quantità e il sistema di esplorazione è sempre lo stesso ma per ogni articolo raccoglie, tra quelli che lo citano, solo il primo risultato. Il grafo risultante avrà l'aspetto di una lunga catena e verrà approfondito nel capitolo successivo.

4.4.2 Produzione del grafo dello stato dell'arte

La produzione del grafo dello stato dell'arte si ottiene principalmente incrociando i risultati delle ricerche effettuate su articoli decisi in modo arbitrario dall'utente. Prima di iniziare, l'utente dovrà specificare le informazioni relative ad alcuni articoli su cui verrà effettuata una ricerca; per migliorare la qualità del risultato finale è consigliabile utilizzare sia articoli con un alto numero di citazioni che

articoli recenti non molto citati ma che si considerano rilevanti. Una volta terminate le ricerche, vengono incrociate le informazioni: ad ogni articolo trovato viene assegnato un punteggio che sarà tanto più alto quanto più spesso questo compare nei grafi delle ricerche, l'idea è che gli articoli che compaiono più frequentemente saranno quelli tenuti maggiormente in considerazione dalla comunità scientifica. Viene anche tenuto conto del numero di citazioni, infatti articoli che compaiono una sola volta ma con numero di citazioni molto più alto della media, verranno comunque considerati rilevanti. Le pubblicazioni selezionate vengono poi inserite in un nuovo grafo con le rispettive relazioni di citazione, così da evidenziarne i legami. Da questo grafo è anche possibile estrarre informazioni come gli autori che hanno contribuito maggiormente allo stato dell'arte.

Dal momento che il grafo dell'arte risulta spesso essere fortemente connesso, quasi completo, è stata aggiunta la possibilità di stampare una lista contenente i paper appartenenti allo stato dell'arte. In questo formato il contenuto è più fruibile per l'utente, in quanto i titoli degli articoli sono più facilmente leggibili da terminale che dentro i nodi del grafo.

4.4.3 Produzione del grafo dei topic

Grazie ai metadati forniti da Academic su campi di studio, è possibile ricavare delle informazioni sul legame tra questi topic. Partendo sempre da un grafo di ricerca, vengono effettuate delle modifiche (vengono aggiunti nuovi archi e nodi) che permettono di estrarre informazioni utili. Le operazioni necessarie sono sostanzialmente tre: all'inizio è necessario creare un nuovo arco tra ogni coppia di articoli legata da una relazione di citazione e i cui articoli possiedono lo stesso campo, la relazione avrà come proprietà il nome del campo in comune; successivamente viene creato un nodo per ogni articolo; infine ogni coppia di nodi verrà legata da un arco se esistono 2 relazioni consecutive (quindi tra tre nodi) create al passo 1 che contengono i nomi di quei campi. L'idea di questo metodo consiste nel fatto che se un articolo che tratta di un topic ne cita un altro che tratta di un topic diverso, questi due campi saranno in qualche modo correlati.

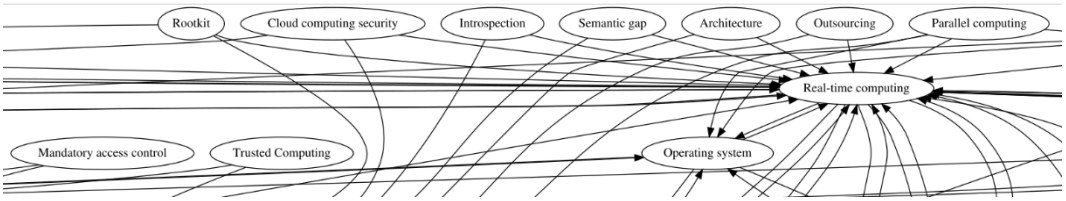


FIG. 10: Porzione del grafo di relazione dei campi di studio

Questa porzione di grafo ne rappresenta tutto il contenuto, grafo si può infatti notare la scarsa quantità di topic con un elevato numero di archi entranti, cioè quelli che compaiono più spesso negli articoli (in questo caso "Real-time computing"), mentre la maggior parte è collegata solo a un paio di quelli principali tramite archi uscenti, tipicamente presenti nella parte superiore del grafo. Questo permette di scremare i campi di studio più rilevanti e diffusi da quelli di nicchia ma in modo sempre relativo a una ricerca specifica, mai in generale in quanto il grafo viene prodotto grazie alle informazioni raccolte da una singola ricerca. È anche interessante notare come i topic meno popolari non siano quasi mai in relazione tra loro; questo non significa che non compaiano spesso ma che gli articoli che li trattano non sono in relazione (non si citano). Altro fattore che permette di identificare un campo di nicchia è l'assenza di archi entranti che indica la mancanza di citazioni da parte di altre pubblicazioni.

È inoltre possibile realizzare un grafo che mostra le relazioni tra gli articoli che trattano lo stesso argomento. Dal grafo di ricerca vengono estratti tutti gli articoli che trattano di un particolare topic con i relativi archi. In questo nuovo grafo è possibile constatare, non solo il numero di articoli che trattano quel particolare argomento ma anche come questi siano legati tra loro. Poche relazioni di citazione indicano che l'argomento non è molto diffuso o lo è solo all'interno di contesti molto ben definiti. Al contrario un grafo ricco di questi archi porta a pensare che quell'argomento. Indipendentemente dal grafo di ricerca si è notato come per alcuni campi di studio (quelli più diffusi), nel loro grafo fossero sempre presenti articoli sia con alto numero di citazioni, ma soprattutto collegati tra loro. Mentre i grafi dei topic meno popolari sono tipicamente composti da pochi articoli isolati; quelli più diffusi possono anch'essi avere pochi nodi, ma questi sono sempre ricchi di archi e il grafo risulta essere fortemente connesso.

5 Risultati Sperimentali

Una delle prime forme di ricerca consisteva nel prendere sempre il primo articoli tra quelli presenti nella pagina degli articoli citanti, questo portava alla produzione di grafi di ricerca con grado molto alto e debolmente connessi.

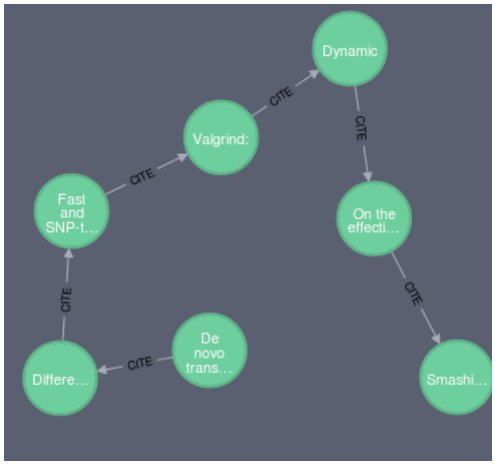


FIG. 11: Esempio di ricerca in profondità

Successivamente, quando è stato realizzato il grafo dello stato dell'arte, è stato interessante notare come in media circa il 70% degli articoli presenti, comparissero anche nei grafi creati con questa tecnica a partire dagli articoli che hanno dato origine al grafo dello stato dell'arte. È da notare che la dimensione dei grafi realizzati con questo metodo non è mai grande in quanto dopo poche iterazioni (tipicamente una quindicina) gli articoli iniziano a ripetersi e il grafo smette di crescere. Questa caratteristica è dovuta al fatto che viene preso sempre l'articolo con più citazioni, che all'interno di un gruppo di articoli tende a essere sempre lo stesso.

Altro metodo di ricerca sviluppato inizialmente ma poi abbandonato, consisteva nel prendere sempre lo stesso numero di articoli citanti da ogni pubblicazione fino a raggiungere una certa quantità. Questo portava alla creazione di grafi poco profondi e ricchi di articoli poco rilevanti.

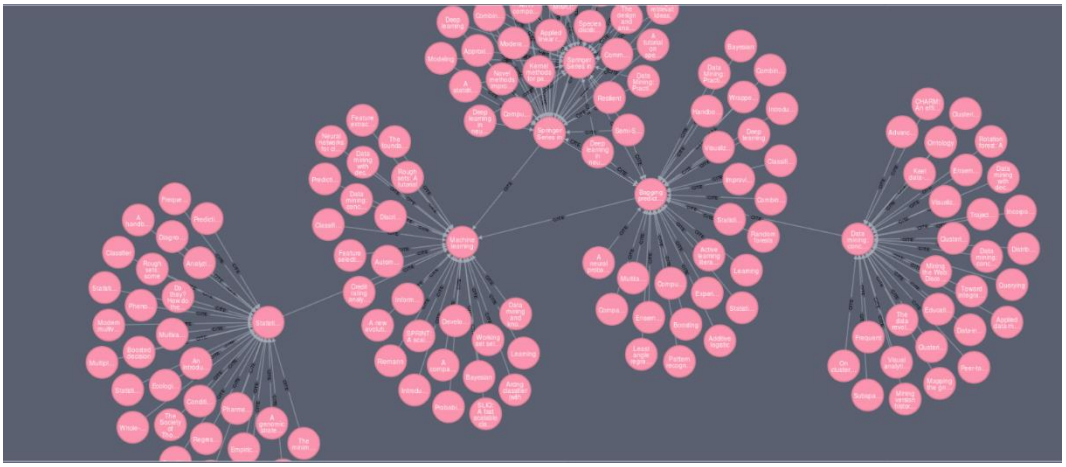


FIG. 12: Esempio di ricerca con numero fisso di elementi

Si è quindi scelto di filtrare le pubblicazioni in base al numero di citazioni. La condizione da soddisfare consisteva di due requisiti: il primo era una soglia sopra la quale questo parametro sarebbe dovuto stare, inizialmente si era pensato a un valore intorno al 1000 ma, in particolare per gli argomenti meno diffusi questo valore si è rivelato troppo alto, si è quindi deciso di abbassarla a 500. Come secondo requisito è stato imposto che il numero di citazioni avrebbe dovuto essere almeno una certa percentuale di quello dell'articolo più citato, anche qui si è partiti con valori abbastanza alti (90%) ma dal momento che spesso già nelle prime pubblicazioni questo valore si abbassa notevolmente, si è deciso di abbassare la percentuale al 50%. Per limitare ulteriormente il numero di articoli raccolti è stato impostato una quantità massima di articoli collezionabili per ogni ricerca, ricordando che l'esplorazione delle citazioni di ogni articolo è assegnata a una routine diversa e che quindi non avviene in modo sequenziale. Questo permette ad articoli, che sequenzialmente verrebbero esplorati molto più avanti nella ricerca, di avere la possibilità di essere raccolti.

Nel motore di ricerca di Google è presente la possibilità di vedere gli **articoli correlati**, oltre a quelli che citano. Sarebbe stato interessante studiare anche questa caratteristica del search engine in quanto poteva essere utile per la costruzione del grafo dello stato dell'arte. Tuttavia, è stato deciso di non considerarla perché Google non fornisce alcuna spiegazione dei criteri che utilizza per decidere se due articoli siano o meno correlati. L'unica spiegazione riguardo questa funzionalità spiega che permette di trovare articoli correlati con i risultati della ricerca.

Gli articoli da cui iniziare la ricerca dovrebbero essere sia popolari (alto numero di citazioni) che **recenti** così da avere una visione globale dell'ambito di ricerca. Gli articoli popolari portano alla scoperta di altri popolari più o meno quanto loro ma spesso si tratta di articoli datati, che noti e riconosciuti da tutta la comunità scientifica; tornerebbe utile utilizzare articoli più recenti, anche con basso numero di citazioni, ma che si ritiene siano rilevanti per l'argomento in questione. L'utilizzo di tali pubblicazioni permetterà di trovare un maggior numero di articoli recenti così da avere una visione migliore della letteratura scientifica.

Durante la realizzazione del software necessario ad incrociare i risultati delle varie ricerche, è stata esplorata la possibilità di **normalizzare il numero di pubblicazioni** così da permettere di confrontare in modo equo due articoli pubblicati a grande distanza di tempo uno dall'altro: l'dea è che a parità di citazioni, un articolo più recente dovrebbe essere considerato con maggiore importanza. Durante lo sviluppo sono state fatte alcune considerazioni: il numero di citazioni cambia enormemente da un campo all'altro e non si riesce a trovare un metro di paragone unico. È stata tentata una valutazione che tenesse conto dei topic ma presentava molte problematicità. Innanzi tutto, i metadati a disposizione: su Academic a un articolo vengono assegnati più topic, risulta quindi difficile normalizzare il numero di citazioni in quanto non è semplice capire quali topic abbiano maggiormente influenzato la diffusione di quell'articolo; su Scholar non è presente un metadato che indichi l'argomento/i trattato dalla pubblicazione e non è quindi possibile alcuna normalizzazione. Anche supponendo che tutti gli articoli trovati appartengano allo stesso campo di ricerca o che sia possibile stabilire il "topic dominante" di un articolo in modo automatico, non sono stati trovati metodi efficaci di valutazione. L'idea era quella di trovare per i topic più rilevanti di una ricerca (tipicamente la top 5 della classifica) e per alcuni anni l'articolo con più citazioni, così da avere un metro di paragone. Le difficoltà insorte hanno riguardato la fase di ricerca per anno, che su Academic si è rilevata molto tediosa in quanto la selezione della data non è semplice da gestire tramite il web driver. Altro problema era dato dalla quantità di dati necessari per completare una ricerca, infatti dopo aver raccolto gli articoli era necessario effettuare una seconda esplorazione per raccogliere le informazioni sui topic; come effetto collaterale questa fase aumenta

di molto il numero di richieste che devono essere effettuate al sito web, ciò si traduce in tempi di attesa molto lunghi (anche diversi minuti per il caricamento di una pagina) che possono portare il web driver a chiudere la connessione prematuramente.

Durante la fase di raccolta di informazioni dai motori di ricerca, è stato affrontato il problema relativo ai **sistemi anti-scraping** messi in atto da questi ultimi verso i web driver. Nelle fasi iniziali dello sviluppo questo era un problema molto marginale, in quanto le ricerche erano brevi e riguardavano poche pubblicazioni, inoltre venivano effettuate a distanza di diversi giorni. Quando si è passati ad effettuare più ricerche (anche 7 o 8) in successione che coinvolgevano anche 300 pubblicazioni ognuna, si sono riscontrate le prime difficoltà. In particolare, su Google Scholar dove, dopo un certo numero di richieste, scattava il controllo reCAPTCHA; al contrario su Microsoft Academic sono stati riscontrati solo dei rallentamenti nelle risposte inviate dal sito. Verso la fine della fase di sviluppo i controlli si sono fatti molto più stringenti: prima era infatti possibile ingannare il controllo aspettando un tempo di attesa casuale (pochi secondi) seguito, ogni tot richieste, da un'attesa molto più lunga (dell'ordine di una decina di minuti); questo sistema rallentava notevolmente la raccolta delle informazioni ma permetteva di mantenere una connessione attiva per ore. Nell'ultimo periodo Google ha introdotto un nuovo sistema di sorveglianza molto più preciso del precedente (27) che non si è riuscito a ingannare e che consente un numero molto limitato di richieste per ip all'ora. Questo ha reso il software molto meno performante ed efficace.

Per gestire i tempi di attesa di Microsoft Academic, che spesso hanno superato i 5 minuti, è stato necessario apportare delle modifiche al software. In particolare, sono state modificate alcune impostazioni del web driver così da non fargli interrompere la connessione; questo ha permesso di produrre grafi di ricerca più gradi e più ricchi di significato. Tuttavia, dopo il suo ultimo aggiornamento anche lui ha introdotto un sistema di controllo contro lo scraping che non sembra bloccare l'utente ma rende ogni tanto il servizio non disponibile e questo peggiora notevolmente la ricerca.

6 Conclusioni

Il progetto soggetto della tesi è nato con l'obbiettivo di creare semplici strumenti che permettessero ad un ricercatore di informarsi in modo più approfondito e completo rispetto a quanto non fosse possibile in precedenza. A tale scopo sono stati creati applicativi che permettessero di raccolte informazioni sugli articoli, di conservare queste informazioni in un database e successivamente di analizzarle. Questo ha permesso di realizzare funzionalità quali: classifiche degli autori più prolifici; grafi che aiutassero nella fase di creazione dello stato dell'arte, proponendo articoli considerati rilevanti per un dato argomento e altri che mettessero in luce le relazioni che esistono tra campi di ricerca affini. E' stato inoltre possibile studiare il comportamento e le operazioni offerte dai motori di ricerca, nonché dei database basati sui grafi.

Pensando a un possibile sviluppo futuro di questa applicazione, la possibilità di confrontare gli articoli ricavati da Google e da Microsoft; infatti al momento non è possibile capire se due entità che provengono da fonti diverse siano effettivamente lo stesso articolo. Incrociare i risultati dei documenti rilevanti tra Google Scholar e Microsoft Academic, permetterebbe di migliorare la qualità e l'esattezza del grafo dello stato dell'arte, in quanto si potrebbero confrontare i grafi di ricerca così da trovare articoli presenti in un motore di ricerca e non nell'altro. Il semplice confronto dei titoli degli articoli non è sufficiente in quanto sono capitati articoli con lo stesso titolo ma con diverso contenuto (anche all'interno dello stesso search engine) e articoli con titoli diversi ma che di fatto corrispondevano alla stessa pubblicazione. Una soluzione potrebbe essere confrontare anche gli autori (che però non sono completamente presenti su Scholar) e la data, così da avere più elementi per paragonare due articoli. Purtroppo, come ulteriore problema le date sono spesso diverse tra i due motori di ricerca e lo stesso articolo è presente più volte con date diverse.

Per velocizzare la ricerca sarebbe utile studiare il comportamento dei motori di ricerca al fine di capire quale comportamento è meglio adottare per evitare rallentamenti da parte di Microsoft Academic, la cui lunghezza a volte provoca il fallimento della ricerca, in quanto non si caricano gli elementi della pagina da cui

vengono estratte le informazioni. Nonostante per Scholar si sia riuscito a "ingannare" il meccanismo che aziona il reCAPTCHA introducendo elementi di casualità, rimane il problema del numero massimo di richieste effettuabili da un utente. Questo limita di molto la quantità di articoli che si riescono ad estrarre, comunque sufficiente per scopi didattici ma non più adeguata quando l'ambito di ricerca è particolarmente vasto o si desidera una migliore precisione nella scelta degli articoli che andranno a comporre lo stato dell'arte (una migliore precisione è data da un numero maggiore di ricerche). Come ultima nota, non è da escludere che, data la velocità con cui Google migliora i propri sistemi, il piccolo elemento di casualità potrebbe non essere sufficiente per evitare di far scattare il controllo; sarà quindi necessario studiare un nuovo sistema per eludere il motore di ricerca.

In conclusione, mi sembrava giusto inserire un ultimo commento sulle funzionalità che riguardano la creazione delle classifiche (sui topic e sugli autori) e in parte anche sul grafo dei campi di studio. Qualche settimana prima della consegna della tesi Microsoft ha rinnovato completamente il look del proprio motore di ricerca e ha introdotto nuove funzionalità. Ora fornisce molte più informazioni sui topic come la classifica degli autori e giornali più influenti, anche gli articoli sono stati arricchiti di metadati; il tutto presentato con grafici che rendono le informazioni molto intuitive e fruibili. Se da un lato ciò ha permesso di valutare con maggiore precisione la qualità del lavoro svolto, dall'altro lo ha reso molto meno utile; in quanto per avere le stesse informazioni (spesso più precise) è sufficiente andare sul sito del motore di ricerca. Tuttavia non fornisce ancora la possibilità di costruire uno stato dell'arte di un dato argomento e, nonostante mostri il topic padre e quelli figli (di un dato topic), questi vengono mostrati solo come una lista di valori invece che in un grafo che ne mostri le relazioni reciproche.

Bibliografia

1. Mestiere di Scrivere.
http://www.mestierediscrivere.com/articolo/articolotecnico.html

2. Università di Verona: Dipartimento di Culture e Civiltà.
http://www.dcuci.univr.it/documenti/OccorrenzaIns/matdid/matdid007359.pdf

3. Wikipedia: Stato dell'Arte.
https://it.wikipedia.org/wiki/Stato_dell%27arte

4. Growth of Science Pubblications.
http://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=5313&context=libphilprac

5. Repository Applicazione.
https://github.com/return55/tirocinio

6. IEEE Explorer. https://en.wikipedia.org/wiki/IEEE_Xplore.

7. Web Scraping. https://en.wikipedia.org/wiki/Web_scraping#DOM_parsing

8. Difference Between Scraping and Crawling.
https://stackoverflow.com/questions/4327392/what-is-the-difference-between-web-crawling-and-web-scraping

9. Comparison Search Engine.
https://dspace3-labs.atmire.com/bitstream/handle/123456789/7634/338.pdf

10. Google Scholar's Ranking Algorithm.
https://www.gipp.com/wp-content/papercite-data/pdf/beel09.pdf

11. Ike Antkare.
http://rr.liglab.fr/research_report/RR-LIG-008.pdf

12. Topics, Microsoft Academic.
https://academic.microsoft.com/#/topics/0/

13. Drahomira Herrmannova, Petr Knoth. An Analysis of the Microsoft Academic Graph.
http://www.dlib.org/dlib/september16/herrmannova/09herrmannova.html

14. Academic, Preview Microsoft.
https://preview.academic.microsoft.com/home

15. Selenium.
https://en.wikipedia.org/wiki/Selenium_(software)

16. Selenium Standalone Server Download.
https://www.seleniumhq.org/download/

17. W3C. WebDriver.
https://www.w3.org/TR/webdriver1/

18. Geckkodriver Releases.
https://github.com/mozilla/geckodriver/releases

19. Driver Selenium Golang.
https://github.com/tebeka/selenium

20. DGraph Github.
https://github.com/dgraph-io/dgraph

21. DB-Engines Ranking of DBMS.
https://db-engines.com/en/ranking/graph+dbms

22. Apoc Documentazione GraphML.
https://neo4j-contrib.github.io/neo4j-apoc-procedures/#graphml

23. Doc Golang Neo4j Driver.
https://godoc.org/github.com/johnnadratowski/golang-neo4j-bolt-driver

24. johnnadratowski. Golang Neo4j Driver.
https://github.com/johnnadratowski/golang-neo4j-bolt-driver

25. Go Documentazione.
https://golang.org/doc/

26. Go Wikipedia.
https://en.wikipedia.org/wiki/Go_(programming_language)

27. Osservazioni Google Scholar max numero richieste.
https://stackoverflow.com/questions/22657548/is-it-ok-to-scrape-data-from-google-results

28. WebDriver Advantages.
https://www.softwaretestingclass.com/what-is-selenium-webdriver-selenium-training-series/

29. Go Installation.
https://golang.org/doc/install

30. Xvfb Debian.
https://packages.debian.org/sid/xvfb

31. Confronto Go e Java.
https://www.educba.com/go-vs-java/