Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

BEX:
un ambiente user-friendly per esplorare articoli scientifici e bibliografie

Relatore:
Chiar.mo Prof.
FABIO VITALI

Presentata da:
RAFFAELE GIANNELLA

Correlatore:
Dott.
ANGELO DI IORIO

Sessione III
Anno Accademico 2013/2014

Indice

Elenco delle figure

Introduzione

Il ricercatore universitario spesso ha la necessità di consultare articoli scientifici che trattino determinati argomenti, generalmente con l'obiettivo di approfondire lo studio di una certa area di ricerca.
Tale attività può limitarsi alla semplice lettura o può delinearsi come un'analisi più approfondita nel caso in cui il ricercatore si stia documentando su tematiche affini a quelle da lui affrontate nella scrittura di un suo articolo; egli potrebbe anche avere l'esigenza di valutare il contesto scientifico legato ad un particolare articolo, ad esempio per fini di revisione.
Per svolgere queste operazioni, oggigiorno, gli strumenti più utilizzati dai ricercatori sono sicuramente rappresentati dalle *Digital Libraries*, cioè servizi disponibili online che consentono di accedere a vaste collezioni di dati e metadati relativi a pubblicazioni scientifiche[1]; la definizione di Digital Library, in realtà, è molto più articolata e complessa, per questo le verrà dedicata un'apposita sezione (sez. 1.1).
Dal confronto diretto con alcuni ricercatori e studiando personalmente alcune digital libraries è emerso che esse, pur essendo strumenti molto completi e potenti, rendono poco efficiente l'esecuzione di determinate attività ed in alcuni casi presentano limitazioni dal punto di vista funzionale. In particolare, ci si riferisce alle attività che per la loro esecuzione prevedano l'analisi della rete citazionale di un articolo, ad esempio: valutare le sue autocitazioni, cioè gli articoli presenti in bibliografia che presentano almeno un autore in

[1] in questa trattazione il contesto di analisi è relativo alle digital libraries di articoli scientifici

comune con l'articolo citante; effettuare osservazioni sull'impatto che esso ha avuto sulla comunità scientifica; analizzare rapidamente alcuni aspetti degli articoli citati, come l'anno di pubblicazione o i principali argomenti trattati.
La digital library è un servizio che deve rendere possibile l'esecuzione di un ampio numero di *tasks*[2] da parte di utenti con in mente obiettivi anche molto diversi; quindi essa fornisce delle funzionalità *general purpose* adatte all'esecuzione di *tasks* generici. Il compromesso che ne deriva spesso si manifesta nella mancanza di specificità di alcune delle funzionalità offerte. Il ricercatore, utilizzando la digital library, non ha a disposizione strumenti (di ricerca, navigazione e *filtering* delle informazioni) progettati appositamente per l'esecuzione rapida ed efficiente di particolari *tasks* che abbiano come attività cardine l'analisi e lo studio delle bibliografie.
Inoltre, spesso le digital libraries forniscono solo una visualizzazione "piatta" degli articoli scientifici, non c'è possibilità di interagire con i dati mostrati e molte informazioni potenzialmente preziose sono nascoste nel contenuto testuale del singolo articolo il quale viene fornito in formati adatti alla stampa (ad esempio il formato *pdf*) piuttosto che ad una consultazione interattiva per mezzo degli strumenti che le tecnologie digitali mettono a disposizione. Negli ultimi anni si è sviluppato un movimento, il *Semantic Publishing*, il quale, sfruttando appieno le tecnologie del *Semantic Web*, ha come primo obiettivo quello di andare a costruire collezioni di articoli scientifici semanticamente arricchiti: l'intenzione è quella di fornire metadati che rendano facilmente accessibili le informazioni generiche relative ad un documento (autori, anno di pubblicazione, etc...) ma soprattutto rendano esplicito e formalizzato il suo contenuto e permettano di stabilire collegamenti con altri articoli semanticamente correlati in modo da promuovere nuove forme di fruizione dei documenti digitali.
A tali fini, sono stati pubblicati diversi dataset, come ad esempio *Nature Pu-*

[2]una sequenza di attività

blishing Group LOD Platform[3], *DBLP++*[4] e *Semantic Lancet Triplestore*[5]. Quest'ultimo dataset, *Semantic Lancet Triplestore*, fornisce informazioni dettagliate e semanticamente arricchite relative alla rete citazionale di un ampio numero di articoli scientifici[6]. Per rete citazionale si intende l'insieme di collegamenti che si vengono a creare tra gli articoli attraverso il meccanismo delle citazioni: un articolo "X", nel suo testo, menziona un articolo "Y" e tra questi due documenti si viene a creare una relazione. Avere le informazioni su queste relazioni è utile, ma parziale; sarebbe interessante avere un maggiore dettaglio informativo sulle citazioni, ad esempio: "perchè un articolo ne cita un altro?", "in che contesto viene citato un certo articolo? quale frase nell'articolo A parla dell'articolo B?". Questi dettagli, che nella maggior parte delle digital libraries non sono facilmente individuabili e accessibili, sono invece disponibili nel *Semantic Lancet Triplestore*, il quale rende espliciti motivi e contesti citazionali e li espone come unità separatamente processabili e analizzabili. Uno studioso che abbia l'esigenza di esaminare nel dettaglio bibliografie e navigare attraverso la rete citazionale di più articoli potrebbe eseguire tali attività in modo molto più efficiente e proficuo se avesse a disposizione le informazioni di cui si è parlato. Ma rendere accessibili tali informazioni strutturate non basta, fornire i dati rappresenta solo il primo passo: il ricercatore può trarre vantaggio da questi dati solo se essi sono veicolati attraverso un'interfaccia semplice da comprendere e da usare. Il secondo passo fondamentale consiste quindi nello sviluppare servizi evoluti che permettano ai ricercatori di fruire di tali contenuti senza la necessità di confrontarsi con la complessità del modello di dati sottostante. Ma ciò non è sufficiente: affinchè tali servizi siano effettivamente di supporto, è essenziale comprendere le esigenze del ricercatore, i suoi obiettivi e l'insieme di attività che egli ha in mente per raggiungere tali obiettivi. Una volta formalizzati

[3]http://www.nature.com/developers/documentation/linked-data-platform/

[4]http://dblp.l3s.de/dblp++.php

[5]http://www.semanticlancet.eu/

[6]attualmente il dataset descrive tutti gli articoli pubblicati presso il Journal of Web Semantics (Elsevier): http://www.journals.elsevier.com/journal-of-web-semantics/

i *goals*[7] e i *tasks* bisogna progettare un'interfaccia che fornisca un insieme di funzionalità ben integrate e coerenti che permettano di assecondare il ricercatore, seguendolo nell'esecuzione delle sue attività; non è l'utente che si adatta all'interfaccia ma è quest'ultima che viene progettata appositamente per adattarsi alle aspettative dell'utente.
Questo è l'obiettivo di BEX, l'applicazione web realizzata per questo progetto di tesi. Essa si configura come un livello intermedio tra il ricercatore e la digital library, fungendo da mediatore tra la complessità dei dati esposti dal modello sottostante e l'esigenza di semplicità d'uso auspicata dall'utente. L'intenzione è quella di fornire al ricercatore un'interfaccia user-friendly attraverso la quale eseguire in maniera rapida ed efficiente delle attività di ricerca, navigazione e valutazione che prevedano lo studio della rete citazionale relativa ad uno o più articoli; ciò è reso possibile grazie alla presenza, nel *Semantic Lancet Triplestore*, di dati semanticamente arricchiti. In seguito a queste attività di analisi e valutazione, l'utente raccoglie un insieme di riferimenti utili e, sempre per mezzo di BEX, ha la possibilità di passare alla digital library che detiene materialmente il contenuto digitale[8] per poter eseguire studi più approfonditi.
BEX è uno servizio sviluppato utilizzando le più moderne tecnologie disponibili nel campo del *web development* e del *web design*, con lo scopo di fornire uno strumento che risulti in linea con principi di buona progettazione, usabilità ed esperienza utente proposti nel mondo delle tecnologie web. Questa applicazione, inoltre, è stata appositamente progettata per renderne facile l'estensione: BEX attualmente fornisce un certo insieme di funzionalità che permettono di sfruttare appieno le potenzialità del dataset sottostante, ma questo dataset fa parte di un progetto in continua evoluzione e, con cadenza mensile, vengono rese disponibili nuove informazioni utili e ben strutturate; sarebbe quindi risultato limitante e controproducente realizzare un'applicazione che si fosse cristallizzata sullo stato attuale delle informazioni dispo-

[7] obiettivi
[8] digital library che detiene i diritti di distribuzione

nibili. Dunque, nelle fasi di progettazione e sviluppo sono stati seguiti dei principi e delle accortezze per rendere possibile l'implementazione di nuove funzionalità senza la necessità di dover riprogettare l'intera applicazione o parti sostanziose della stessa.

Per raggiungere i risultati illustrati è di primaria importanza comprendere cosa sia effettivamente una Digital Library, analizzarne le funzionalità per metterne in evidenza i tratti distintivi, facendo emergere i punti di forza e di debolezza: l'analisi condotta è illustrata nel Capitolo 1.

Nel Capitolo 2 vengono descritti gli ideali e le tecnologie del Semantic Web e del Semantic Publishing, concetti che costituiscono le fondamenta del *Semantic Lancet Project*, il progetto universitario nel quale si va ad innestare questo lavoro di tesi.

Il Capitolo 3 è dedicato all'analisi di BEX, dei principi di *User Interface Design* e *User Experience* che hanno guidato la progettazione e l'implementazione delle funzionalità offerte dall'applicazione; tali funzionalità vengono poi, ovviamente, descritte nel dettaglio.

Il Capitolo 4 è dedicato ai dettagli implementativi: gli obiettivi di progettazione, i *design patterns* applicati, i principali frameworks e librerie utilizzati durante lo sviluppo per dare all'applicazione una struttura interna coerente e ben organizzata; infine verranno brevemente descritti moduli e componenti principali del progetto.

Il Capitolo 5 presenta alcune valutazioni relative alle modalità di recupero delle informazioni per poi passare ad una breve analisi dei primi test effettuati con gli utenti finali.

Capitolo 1

Le digital libraries e l'efficacia di strumenti di supporto al loro utilizzo

Al fine di individuare e caratterizzare le funzionalità fondamentali comuni alle digital libraries accessibili sul web è stata condotta un'analisi su tre delle principali e più note biblioteche digitali di pubblicazioni scientifiche: IEEEXplore[1], ScienceDirect [2] e ACM[3]. Tale analisi si è rivelata fondamentale per la comprensione delle modalità di navigazione e fruizione di contenuti digitali caratterizzati da un elevato carico informativo; in seguito, lo studio di come le informazioni vengano presentate all'utente ha permesso di determinare quali siano gli aspetti critici che determinano la differenza tra una buona esperienza utente ed una mediocre.
Si ritiene utile partire da una definizione di cosa sia una digital library per poi passare alla descrizione dei punti fondamentali dell'analisi condotta.

[1]http://ieeexplore.ieee.org/
[2]http://www.sciencedirect.com/
[3]http://dl.acm.org/

1.1 Cos'è una digital library

Dare una definizione precisa e puntuale di cosa sia una Digital Library (anche definita Biblioteca Digitale) non è compito semplice: le caratterizzazioni sono molteplici, la letteratura fornisce numerose definizioni le quali variano in base all'ambito di formulazione; alcune sono legate a specifici progetti di realizzazione di digital library, altre sono riconducibili al mondo della ricerca e riflettono le diverse interpretazioni del fenomeno della biblioteca digitale. Una prima definizione è stata data da Christine Borgman nel 1993 [BOR93] che indicò la biblioteca digitale come combinazione di:

- un servizio
- un'architettura di rete
- un insieme di risorse informative comprendenti dati testuali, numerici, immagini, suoni, video, etc...
- un insieme di strumenti per localizzare, recuperare e utilizzare queste risorse informative

Secondo una definizione data da William Y. Arms [ARM00] la biblioteca digitale è costituita da una collezione di informazioni, organizzata insieme a servizi correlati, dove l'informazione considerata è in formato digitale e i servizi sono accessibili attraverso la rete.
Un'ulteriore caratterizzazione è data da Oppenheim e Smithson [OS99] che pongono l'attenzione sull'utilizzo delle tecnologie digitali: la biblioteca digitale è un servizio informativo, in cui tutte le risorse informative sono disponibili in formato digitale e tutte le funzioni di acquisizione, archiviazione, preservazione, recupero ed accesso sono realizzate per mezzo di tecnologie digitali. In ambito bibliotecario, una definizione rilevante è quella della Digital Libraries Federation (DLF) [4] "Le biblioteche digitali sono organizzazioni che forniscono le risorse, compreso il personale specializzato, per selezionare, organizzare,

[4] http://www.clir.org/dlf

dare l'accesso intellettuale, interpretare, distribuire, preservare l'integrità e assicurare la persistenza nel tempo delle collezioni digitali così che queste possano essere accessibili prontamente ed economicamente per una comunità definita o per un insieme di comunità".
In Italia, per lungo tempo, si è utilizzato il termine "biblioteca virtuale" per definire il concetto di biblioteca digitale; il primo ad usare il termine "biblioteca virtuale" (Virtual Library) è stato lo stesso autore del web, Tim Berners-Lee: egli si è occupato della creazione di un portale[5] con l'intenzione di realizzare una visione di biblioteca come una vasta collezione di documenti collegati in rete, costituiti da oggetti digitali e pagine web realizzate da migliaia di autori. Il termine "biblioteca digitale", in Italia, si afferma verso la fine degli anni '90; il termine è introdotto nel 1998 da Malinconico "Le tecnologie digitali facilitano l'accesso alle raccolte bibliotecarie, trasferendo i contenuti delle fonti d'informazione o loro fedeli rappresentazioni attraverso lo spazio, dal luogo in cui sono conservate a quello in cui sono richieste. Le stesse tecnologie potrebbero essere utilizzate per trasportare la sostanza dei materiali nel tempo, contribuendo in tal modo alla loro conservazione" [MAL98]. Un approfondimento rilevante arriva nel 2002 con Ciotti e Roncaglia che parlano dell'organizzazione dei documenti e dei metadati: "definiamo "biblioteca digitale" una collezione di documenti digitali strutturati (sia prodotti mediante digitalizzazione di originali materiali, sia realizzati ex-novo), dotata di un'organizzazione complessiva coerente di natura semantica e tematica, che si manifesta mediante un insieme di relazioni interdocumentali e intradocumentali e mediante un adeguato apparato metainformativo. In questo senso possiamo distinguere una biblioteca digitale da un insieme non organizzato di informazioni assolutamente eterogenee come World Wide Web, ma anche da molti archivi testuali che attualmente sono disponibili su Internet e che si presentano come "depositi testuali" piuttosto che come vere e proprie biblioteche" [CR02].

[5] http://vlib.org/

1.2 Esempi di Digital Libraries moderne

Le digital libraries[6] oggi disponibili sono strumenti molto evoluti e permettono l'accesso a banche dati di grandi dimensioni: vaste collezioni di documenti digitali, in particolare, articoli pubblicati da riviste legate alla comunità scientifica.

I metadati

Il valore aggiunto delle informazioni disponibili è costituito dai cosiddetti metadati: informazioni strutturate di diversa tipologia, interpretabili da parte di un computer (*machine-readable*), associati ad una risorsa per specificarne delle proprietà.
Relativamente alle Digital Libraries, di particolare interesse sono due classi di metadati:

- **metadati descrittivi** (o metadati per *resource discovery*): utilizzati per descrivere ed identificare le risorse costituenti la biblioteca digitale, allo scopo di fornire funzionalità di ricerca. Esempi di metadati descrittivi relativi ad un documento: titolo, autori, anno di pubblicazione, tipologia di articolo.

- **metadati strutturali**: atti a descrivere la struttura delle risorse di una Digital Library; essi permettono inoltre di descrivere le relazioni tra le risorse o tra le parti di una stessa risorsa. Esempi di metadati strutturali relativi ad un documento: i riferimenti tra documenti (citazioni), suddivisione del documento (capitoli, sezioni, sottosezioni)

La ricerca e i risultati

Grazie alla presenza di questo livello di "informazioni sulle informazioni", le digital libraries permettono di integrare strumenti di ricerca molto avanzati ed efficienti. Solitamente gli utenti possono effettuare ricerche generiche

[6] vengono prese in considerazione DL di pubblicazioni scientifiche

partendo da semplici parole chiave o possono decidere di impostare anche delle opzioni di ricerca.
Alcune comuni opzioni:

- **autori**: filtrare i risultati mostrando solo quelli di determinati autori
- **editore**: mostrare solo gli articoli pubblicati da un certo editore
- **anno di pubblicazione**: filtrare i risultati in base all'anno di pubblicazione degli articoli
- **tipo di articolo**: articoli scientifici, editoriali, articoli di rassegna, etc...
- **settore di riferimento**: informatica, matematica, biologia, medicina, scienze cognitive etc...
- **argomento**: semantic web, intelligenza artificiale, nanotecnologie, etc...

Figura 1.1: Elemento dei risultati di IEEEXplore

Figura 1.2: Elemento dei risultati di ScienceDirect

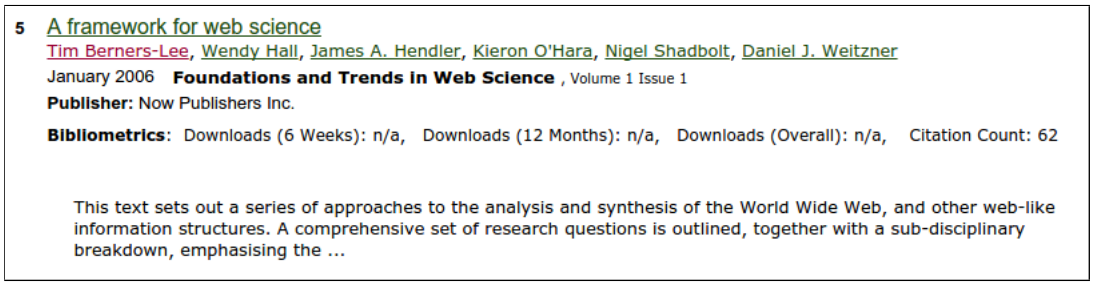
5 A framework for web science
Tim Berners-Lee, Wendy Hall, James A. Hendler, Kieron O'Hara, Nigel Shadbolt, Daniel J. Weitzner
January 2006 **Foundations and Trends in Web Science** , Volume 1 Issue 1
Publisher: Now Publishers Inc.
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count: 62
This text sets out a series of approaches to the analysis and synthesis of the World Wide Web, and other web-like information structures. A comprehensive set of research questions is outlined, together with a sub-disciplinary breakdown, emphasising the ...

Figura 1.3: Elemento dei risultati di ACM-DL

Dopo aver effettuato la ricerca, all'utente viene presentata una lista di risultati i quali tipicamente mostrano le informazioni generiche su ogni articolo.
In Fig. 1.1, Fig. 1.2 e Fig. 1.3 sono riportati esempi di un singolo elemento dei risultati di ricerca, rispettivamente di IEEEXplore[7], ScienceDirect[8] e ACM[9]; come si può osservare, le informazioni fondamentali che vengono fornite per ogni elemento sono abbastanza omogenee: titolo, autori, anno di pubblicazione, editore, volume, issue. Viene inoltre fornita la possibilità di leggere l'abstract dell'articolo ed in alcuni casi sono presenti informazioni generiche sulle citazioni.
A questo livello non sono disponibili altre informazioni; per approfondire l'analisi di un elemento si richiede il passaggio ad altre pagine, ad esempio cliccando sul titolo, su uno degli autori o sull'editore.

Il singolo articolo

Per avere ulteriori informazioni su di un articolo è necessario cliccare sul suo titolo per essere reindirizzati ad un'altra pagina: qui, oltre alle informazioni già visualizzate precedentemente, vengono forniti dettagli riguardanti la bibliografia, le citazioni verso l'articolo e vari indici bibliometrici. Queste informazioni su bibliografia e citazioni sono molto preziose in quanto per-

[7] http://ieeexplore.ieee.org/
[8] http://www.sciencedirect.com/
[9] http://dl.acm.org/

mettono di collegare l'articolo esaminato ad un insieme di altri lavori che gli gravitano intorno, permettendo di fare alcune osservazioni.
La bibliografia ad esempio può essere utile per capire su cosa abbiano lavorato gli autori dell'articolo, se si tratta di tematiche di recente sviluppo e discussione (se cita articoli recenti), se gli autori stanno lavorando molto su delle idee e su un determinato progetto (se ci sono molte autocitazioni).
Le citazioni verso l'articolo consentono di farsi un'idea di come la comunità scientifica abbia reagito all'articolo stesso.

1.3 Considerazioni sulle funzionalità offerte e sull'architettura delle informazioni

Le funzionalità di ricerca e navigazione offerte dalle digital libraries analizzate sono sicuramente complete, permettono di accedere a tutti i contenuti digitali messi a disposizione e di navigare tra essi; tali funzionalità sono orientate alle esigenze di un utente generico che ha degli obiettivi e utilizza il sistema avendo in mente dei tasks da eseguire per raggiungere questi obiettivi.
Per soddisfare le esigenze di un bacino di utenti molto variegato è necessario rendere possibile l'esecuzione di un ampio numero di tasks diversi tra di loro e per permettere ciò gli strumenti messi a disposizione devono risultare flessibili; come evidenziato in [ABO97], gli utenti possono esplorare i contenuti in ogni modo concepibile e l'organizzazione delle informazioni non deve essere distorta dalle aspettative riguardanti come gli utenti potrebbero approcciarsi al materiale, il loro livello di esperienza e competenza: deve quindi essere fornita flessibilità.
Ciò porta inevitabilmente ad un compromesso: si perde in specificità di alcune funzionalità e semplicità di utilizzo.
Questo trade-off va ad influenzare la struttura logica organizzativa e semantica delle informazioni e delle funzionalità fornite dal sistema: è una questione di *Information Architecture*.

1.3.1 Una parentesi su Information Architecture e Sensemaking

L'*Information Architecture* (architettura delle informazioni) svolge un ruolo fondamentale nel definire come le persone processino cognitivamente l'informazione che gli viene fornita; il suo principale obiettivo è quello di gestire i contenuti informativi, organizzarli, strutturarli, ordinarli e presentarli all'utente in modo che egli possa dare loro un senso [GAR10].
I contenuti (dati) di per sé non sono particolarmente utili, hanno un valore, ma diventano rilevanti per l'utente nel momento in cui egli li pone in relazione tra di loro fornendogli un contesto, ottenendo quindi le informazioni; in seguito, mettendo queste informazioni in relazione tra di loro sulla base dell'apprendimento, se ne comprendono il significato ed i modelli sottostanti, passando dunque alla conoscenza: "*information is not knowledge*" - Albert Einstein.
Daniel M. Russell [RSP93] definisce il sensemaking come un processo di ricerca di una rappresentazione e di una codifica dei dati in tale rappresentazione al fine di rispondere alle questioni relative ad uno specifico task; durante l'esecuzione di un task che richiede l'elaborazione di informazioni, le diverse operazioni di sensemaking richiedono diverse risorse cognitive, queste operazioni hanno quindi un costo cognitivo e la scelta di una rappresentazione ha come obiettivo quello di ridurre questo costo.
Una definizione alternativa, in un'ottica socio-psicologica, viene fornita da Karl Weick [WEI96]: il sensemaking riguarda la riconduzione di certi stimoli ad un quadro di senso, a dei contesti (*frameworks*); questi contesti potrebbero essere categorizzazioni, anticipazioni o assunzioni.
Riassumendo, l'attribuzione di senso comporta la raccolta di informazioni, il passaggio alla comprensione di queste informazioni al fine di utilizzare tale comprensione per completare un compito (*task*) [SHA07].
Questo passaggio deve essere spontaneo e non può essere forzato: come renderlo possibile? bisogna facilitare la creazione di contesti e relazioni da parte dell'utente.

Per ottenere questo risultato rendere un'informazione trovabile non basta, è necessario comprendere a fondo i bisogni e le intenzioni degli utenti per arrivare a strutturare l'informazione in modo da favorire la costruzione spontanea di relazioni fra gli oggetti informativi che l'utente si trova ad analizzare, abbassare quindi il costo cognitivo richiesto per l'esecuzione di operazioni di reperimento delle informazioni (*information retrieval*); in questa ottica è fondamentale cercare delle risposte ad alcune domande:

- con che livello di dettaglio mostrare le informazioni?
- quale sistema di navigazione adottare?
- per favorire l'esecuzione dei tasks dell'utente, quali informazioni sono rilevanti?
- come presentare contenuti informativi senza determinare un sovraccarico per l'utente?
- come raggruppare le componenti di informazione in categorie significative e distinte?
- quanti step sono necessari per completare un determinato task? ognuno di questi step ha un senso per l'utente? sono collegati in modo intuitivo tra di loro? [GAR10]

1.3.2 Inefficienze delle Digital Libraries

Alla luce di queste considerazioni sull'*Information Architecture*, tornando alle funzionalità offerte dalle digital libraries, un punto fondamentale della trattazione è costituito dalla riflessione che viene qui di seguito esposta.
Si ponga il caso di un utente più caratterizzato dell'utente generico, con un determinato background culturale ed intellettuale, ad esempio un ricercatore universitario. Si consideri poi un ricercatore che per raggiungere i suoi obiettivi (goals) abbia la necessità di eseguire delle attività che coinvolgano l'analisi di numerose bibliografie al fine di valutare un determinato insieme

di articoli o per navigare attraverso la loro rete citazionale. La digital library sicuramente fornisce al ricercatore delle funzionalità di ricerca e navigazione che gli permettono di arrivare alle informazioni che sta cercando, ma la mancanza di specificità di alcune di queste funzionalità porta a degli inconvenienti.
Vengono ora illustrati alcuni casi generali di problematiche riscontrabili nell'utilizzo delle digital libraries *general purpose*[10] da parte di un ricercatore che esegue un (relativamente) ristretto numero di tasks.
I seguenti casi sono generali e non contestuali ad una DL[11] in particolare; le osservazioni proposte nascono dal loro diretto utilizzo e dal confronto con alcuni ricercatori che utilizzano quotidianamente questi strumenti.
Lo scopo è quello di meglio comprendere il dominio delle problematiche affrontate, le quali hanno in seguito portato alla ideazione e definizione delle funzionalità ritenute utili in funzione di un'esecuzione efficace di determinati tasks da parte dei ricercatori universitari. Per una trattazione più completa e dettagliata sui tasks presi in considerazione si rimanda alla sezione 3.2.

Informazioni sparse

Il ricercatore, nel realizzare il suo task, si ritrova a dover intraprendere un numero di azioni (o steps nella navigazione) maggiore di quello che sarebbe stato richiesto se la funzionalità[12] fosse stata progettata appositamente per l'esecuzione di quel preciso task; questo problema può manifestarsi con un elevato numero di interazioni con il sistema (click, ad esempio) prima di poter arrivare alle informazioni ritenute rilevanti: l'utente deve interagire ripetutamente con gli elementi informativi visualizzati, muovendosi su più pagine durante la navigazione e questo porta inevitabilmente ad un allungamento dei tempi di esecuzione del task con un conseguente calo della soddisfazione per l'utente. Da non sottovalutare, inoltre, che per il passaggio da una pagina

[10] con digital library general purpose si intende una digital library che fornisce funzionalità non specializzate per l'esecuzione di determinati tasks
[11] Digital Library
[12] funzionalità di ricerca, di filtraggio, di navigazione

all'altra si rendono necessarie ulteriori richieste al server per poter caricare i nuovi elementi e questo determina ulteriori rallentamenti (e nervosismi). Il caso tipico è quello dell'utente che, nella lista dei risultati di ricerca, non ha a disposizione alcune informazioni generiche ritenute rilevanti riguardo ad un articolo e perciò si trova a doversi spostare sulla pagina contenente gli ulteriori dettagli sull'articolo di interesse.

Informazioni non aggregate

Il ricercatore, nel realizzare il suo task, esplorando i contenuti digitali ha l'esigenza di avere uno sguardo di insieme su un particolare dominio di dati per poter valutare rapidamente la bontà di un certo contenuto digitale[13], ma le informazioni che la DL fornisce al riguardo sono frammentate, disposte in forma non aggregata o compatta e richiedono che l'utente, per giungere alle sue conclusioni, navighi su più livelli, interagendo ripetutamente con il sistema per ricavare manualmente le informazioni ritenute significative; questo, oltre a presentare gli inconvenienti del caso precedente, sfavorisce l'attività di sensemaking di cui si è discusso in precedenza (sez. 1.3.1) in quanto il reperimento delle informazioni (information retrieval) è intaccato da uno schema di navigazione che costringe l'utente a dover elaborare una grande quantità di informazioni non rilevanti, portando così ad un sovraccarico cognitivo.

Informazioni solo aggregate

Potrebbe sembrare paradossale, ma anche il caso opposto rispetto al precedente, in alcune circostanze (discusse nel Capitolo 3), non aiuta l'utente nell'esecuzione di un task. Il problema fondamentale è la mancanza di profondità informativa: la DL fornisce un'informazione (relativa ad un articolo) in forma aggregata, come ad esempio un contatore o un indice, ma non permette di visualizzare, perlomeno non direttamente, i dati che lo compongono. Questo rende difficile per l'utente approfondire l'analisi di un determinato dominio di dati; molto spesso è richiesto che egli effettui, come nei casi precedenti,

[13] in questo contesto il contenuto digitale potrebbe essere un articolo scientifico

ulteriori passi nella navigazione per ricavarsi indirettamente le informazioni che vanno a comporre l'indicatore aggregato visualizzato in precedenza. Ciò ha evidenti ripercussioni sull'esperienza utente e sul sensemaking.

Informazioni non disponibili

Il fine ultimo di una DL è quello di rendere accessibili, in modo più o meno libero[14], gli articoli scientifici e il loro contenuto testuale affinché l'utente possa studiarli e trarne conoscenza; dopo aver acquisito (in modo gratuito o meno) l'accesso al contenuto dell'articolo scientifico, l'utente ne studia le componenti e il contenuto al fine di raggiungere i suoi particolari obiettivi; è un'attività intellettualmente impegnativa ed in quanto tale richiede un importante investimento in termini di tempo e di risorse cognitive. Questa è un'operazione indubbiamente essenziale per determinati tasks per i quali sia necessaria un'analisi approfondita di un singolo paper, ma l'utente degli esempi precedenti, cioè il ricercatore, quotidianamente si ritrova ad eseguire anche alcuni tasks che prevedono l'analisi di un quantitativo non indifferente di articoli scientifici; per ognuno di questi non è necessario lo studio puntuale di ogni componente (anche perchè sarebbe insostenibile) ma piuttosto, in base all'obiettivo prefissato, è utile concentrarsi prima su alcuni elementi in particolare per poi passare, eventualmente, ad una valutazione dell'articolo preso in considerazione e delle informazioni in esso contenute.
Si prenda ad esempio la bibliografia di un articolo: è una sezione analizzata molto attentamente dal ricercatore nella sua attività di valutazione (i tasks analizzati sono descritti nel dettaglio nella sezione 3.2) in quanto fornisce utili informazioni sull'articolo analizzato e permette ulteriori approfondimenti sulla materia trattata.
Per il ricercatore è molto utile anche sapere il perché un certo elemento della bibliografia viene citato nell'articolo o, inversamente, da chi (da quali articoli)

[14] una pubblicazione potrebbe essere disponibile in Open Access, cioè senza restrizioni, ma potrebbe anche essere richiesto un abbonamento per visualizzare i contenuti protetti da particolari licenze

e perchè (per quali motivi) l'articolo analizzato è citato; un'altra informazione preziosa potrebbe essere il contesto in cui un certo articolo viene citato (contesto citazionale). In altre parole, sarebbe vantaggioso per il ricercatore avere informazioni più approfondite sulla rete citazionale di cui fa parte l'articolo analizzato.
Per come sono strutturate le digital libraries, per i dati che esse espongono e per le funzionalità (di ricerca, navigazione e filtering) messe a disposizione dell'utente, queste informazioni dettagliate sulla rete citazionale non sono disponibili oppure sono "nascoste" nel contenuto testuale del paper e perciò difficilmente accessibili ed individuabili.
Le digital libraries spesso mostrano le bibliografie come "unità monolitiche", gli utenti non hanno la possibilità di analizzare gli elementi singolarmente [DGP15] e non viene fornita alcuna funzionalità di filtering sulla bibliografia. Di conseguenza il ricercatore, nell'esecuzione del suo task, si ritrova ancora una volta ad esplorare in profondità gli elementi dei risultati di ricerca e a dover ricavare alcune informazioni dall'analisi del contenuto testuale degli articoli: operazioni che richiedono molto tempo e che potrebbero essere evitate o facilitate fornendo all'utente, nei punti giusti, delle informazioni ben strutturate.

1.4 L'emergere di esigenze particolari per i ricercatori

I casi analizzati precedentemente prendono come esempio di utente il ricercatore universitario: questo particolare segmento di utenza fa un uso massivo delle digital libraries per l'esecuzione quotidiana di alcuni tasks con obiettivi definiti dal ruolo che ci si trova a ricoprire; egli ad esempio può essere:

- **lettore**: ricerca articoli, ne fa una prima selezione basandosi su pochi indicatori, approfondisce, legge e naviga tra essi muovendosi prevalen-

temente attraverso la rete citazionale

- **autore**: effettua ricerche per trovare lavori che trattino di tematiche vicine a quelle da lui affrontate nella stesura di un suo paper

- **recensore**: nel revisionare uno o più articoli, analizza anche la loro bibliografia, le autocitazioni[15] e gli anni di pubblicazione degli articoli citati

Si viene a delineare una figura di utente con precise esigenze:

- avere la possibilità di muoversi tra gli articoli scientifici per mezzo della rete citazionale

- esaminare informazioni in forma aggregata (ad esempio le citazioni) e, nel caso lo ritenga necessario, avere ulteriori dettagli in merito

- dare una prima valutazione di uno o più articoli partendo da pochi elementi informativi, per poi approfondire nel caso un articolo sia ritenuto rilevante

- avere un accesso facilitato alle informazioni relative agli elementi bibliografici di un articolo

- avere informazioni su come la comunità scientifica abbia reagito ad un certo articolo, per mezzo delle citazioni

- ordinare un determinato set di elementi, come ad esempio i risultati di una ricerca o gli elementi della bibliografia, in base ad un certo parametro (anno di pubblicazione, numero di citazioni, etc...)

- filtrare gli elementi bibliografici per mettere in evidenza solo quelli ritenuti interessanti, basandosi anche in questo caso su un parametro scelto dall'utente

[15] citazioni ad articoli con cui è condiviso almeno un autore

Per soddisfare queste esigenze si rende necessario fornire al ricercatore funzionalità di ricerca, filtering e visualizzazione progettate appositamente per favorire il reperimento rapido di informazioni rilevanti per una prima valutazione su un insieme di risultati e permettere in seguito ulteriori approfondimenti su uno o più articoli, consentendo una navigazione tra gli elementi che risulti il più possibile efficiente.
L'utente non deve essere portato a spostarsi continuamente da una pagina all'altra, ma deve poter visionare in un solo punto le informazioni fondamentali e di interesse per un certo ambito informativo e, nel caso decida di spostarsi, deve essere sempre consapevole di dove si trova e di che tipo di informazioni sta visualizzando (contesto).

1.5 L'utilità di strumenti di supporto all'uso delle digital libraries

Alla luce di quanto detto, è possibile esprimere una considerazione sul perché sia utile l'utilizzo di strumenti, applicazioni e servizi da affiancare alle digital libraries usate quotidianamente: queste ultime non mancano di funzionalità, ma mancano di funzionalità *specifiche* che permettano ad una categoria *specifica* di utenti di eseguire *specifici* tasks in modo rapido ed efficiente.
La digital library come portale di ricerca di contenuti digitali costituisce uno strumento insostituibile, ma può anche risultare macchinoso e poco intuitivo all'utilizzo nel caso in cui i tasks che si vanno ad eseguire prevedano attività che richiedono l'analisi di grandi quantità di informazioni in poco tempo: sono questi i casi in cui si può rivelare di grande aiuto uno strumento di supporto che fornisca delle funzionalità appositamente progettate per l'esecuzione efficiente di determinati tasks e con una architettura delle informazioni che permetta all'utente di risparmiare in termini di tempo e risorse cognitive; questo con l'intenzione di accrescere la soddisfazione dell'utente e favorire l'attività di sensemaking di cui si è parlato in precedenza.

Quindi questo strumento di supporto si configura come un livello intermedio, una sorta di interfaccia che si frappone tra il ricercatore e la DL, fungendo da mediatore tra la complessità dei dati esposti dalla DL e l'esigenza di semplicità d'uso auspicata dal ricercatore.
Nel caso particolare di questo progetto di tesi, questo strumento di supporto si realizza nella forma di un'applicazione web che, per assistere il ricercatore nel raggiungimento di certi obiettivi, si interfaccia con un dataset del tutto particolare per il recupero delle informazioni da riproporre poi all'utente; la struttura di questo dataset è analizzata nella sezione 2.3.3

1.6 La necessità di informazioni strutturate e semanticamente arricchite

Per rendere possibile lo sviluppo di un'applicazione web che risponda ai requisiti analizzati nelle sezioni precedenti è innanzitutto necessario avere accesso ad una ricca e ben strutturata collezione di dati e metadati.
Un primo problema è che le repository dei servizi disponibili online non offrono APIs[16] per accedere a queste informazioni (ad esempio Google Scholar[17]) o espongono APIs che non rendono accessibili tutte le informazioni ritenute necessarie (ad esempio ScienceDirect[18] e Scopus[19])[DGP15]. Allora dove prendere queste informazioni strutturate sugli articoli scientifici e sulla loro rete citazionale? La risposta è: *Semantic Lancet Triplestore*.
Il *Semantic Lancet Triplestore* (SLT) è una collezione di dati e metadati relativi a pubblicazioni scientifiche: in particolare si tratta di un *Linked Open Dataset* liberamente accessibile. Nel prossimo capitolo verrà illustrato cosa sia SLT e come sia internamente strutturato, ma prima di poterne parlare è necessario introdurre ed esporre quali siano gli ideali, le tecnologie, i servizi ed i linguaggi che stanno alla base del progetto: la radice di tutto ciò è il *Semantic Web*.

[16] Application Programming Interface
[17] https://scholar.google.it/
[18] http://www.sciencedirect.com/
[19] http://www.scopus.com/

Capitolo 2

Semantic Web, Semantic Publishing, Semantic Lancet Project

Questo capitolo ha come primo obiettivo quello di illustrare quali siano gli ideali, le tecnologie ed i linguaggi fondamentali che stanno alla base del Semantic Web; in seguito verrà trattato il tema del Semantic Publishing per poi passare ad una prima analisi dettagliata del Semantic Lancet Project e del suo dataset.

2.1 Cos'è il Semantic Web

"The Semantic Web is a web of data", questa è una prima definizione fornita dal W3C[1]. Il semantic web riguarda due aspetti fondamentali [2]:

- **formati comuni** e condivisi per l'integrazione e la combinazione di dati provenienti da fonti diverse
- **linguaggi** per descrivere come questi dati si rapportino agli oggetti del mondo reale

[1] http://www.w3.org/

[2] Introduction to the W3C Semantic Web Activity: http://www.w3.org/2001/sw/

Il fine ultimo del Semantic Web è quello di rendere possibile per le macchine eseguire del lavoro più utile e sviluppare sistemi che supportino interazioni affidabili sulla rete; il semantic web potrà mostrare le sue potenzialità quando gli utenti avranno fiducia nelle sue operazioni e nella qualità delle informazioni fornite.[3].

Tutto si basa sull'ipotesi che le macchine possano accedere ad un insieme strutturato di informazioni e ad un insieme di regole di inferenza utilizzabili per il ragionamento automatico.

Il termine "semantic web" si riferisce alla visione del W3C di un web come linked data; le tecnologie del semantic web permettono alle persone di creare archivi di dati sul web, costruire vocabolari e definire regole per la gestione dei dati.

Tim Berners-lee, ideatore del semantic web, da la seguente definizione: Il Semantic Web è un'estensione dell'attuale Web, nella quale all'informazione viene dato un significato ben definito, permettendo così ai computer e alle persone di lavorare meglio in cooperazione [BHL01].

Il semantic web non è un qualcosa di totalmente diverso dal web tradizionalmente inteso, ne è un'estensione: fornisce modalità standardizzate per esprimere le relazioni tra le pagine web in modo da rendere possibile per le macchine comprendere il significato delle informazioni; nel web semantico, ai documenti pubblicati sono associate ulteriori informazioni, metadati che ne forniscono un contesto semantico.

Una delle sfide principali per la realizzazione del web semantico consiste nel fornire linguaggi e modelli per esprimere dati e regole per ragionare su questi dati: le macchine potrebbero così analizzare le informazioni provenienti da diverse fonti e combinarle per trarne nuova conoscenza. Riguardo all'accesso e all'elaborazione delle informazioni da parte delle macchine (machine-accessibility): "The main obstacle to providing better support to Web users is that, at present, the meaning of Web content is not machine-accessible. Of course, there are tools that can retrieve texts but the capabilities of current

[3]tratto e liberamente interpretato da: http://www.w3.org/standards/semanticweb/

software are still very limited. An alternative approach is to represent Web content in a form that is more easily machine-processable and to use intelligent techniques to take advantage of these representations. We refer to this plan of revolutionizing the Web as the Semantic Web initiative."[AH08]

2.1.1 Costruzione: Resource Description Framework

RDF (Resource Description Framework) è un modello standard per l'interscambio di dati sul web: è lo strumento base per la codifica, lo scambio e il riutilizzo di metadati strutturati e consente l'interoperabilità tra applicazioni che si scambiano sul Web informazioni *machine-understendable* [SIG02]. RDF è costituito da due componenti:

- **RDF Model and Syntax**: definisce il *data model* RDF e la sua codifica XML
- **RDF Schema (RDFS)**: è un linguaggio per creare vocabolari e ontologie usando RDF

RDF Data Model

Alla base del modello ci sono tre tipi di oggetti:

- **Risorsa**: tutto ciò che può essere descritto tramite un'espressione RDF; una risorsa, sostanzialmente, può essere qualsiasi cosa, ad esempio una pagina web, più pagine web, un documento, un'immagine, un video, ma anche un libro, una persona, un luogo, una relazione. L'importante è che ad essa sia associato un Uniform Resource Identifier (URI)
- **Proprietà**: è una caratteristica, un attributo, una relazione usata per descrivere un risorsa. Ogni proprietà ha un nome e definisce un insieme di valori ammissibili. Ogni proprietà è anch'essa una risorsa.

- **Statement**: è una tripla costituita da *Soggetto, Predicato, Oggetto*; il Soggetto è una risorsa, il predicato è una proprietà, l'oggetto è un valore che può essere un'espressione (una sequenza di caratteri o un altro tipo primitivo definito da XML) oppure un'altra risorsa.

Un insieme di più statement rdf si definisce *Grafo RDF*.

Ontologie, RDFS e OWL

RDF Data Model, come indica il nome, permette di definire un modello per descrivere le relazioni tra le risorse, in termini di proprietà identificate da un nome e relativi valori, ma non fornisce alcun meccanismo per la dichiarazione di queste proprietà e non permette di definire le relazioni tra proprietà e altre risorse[SIG02], non è un linguaggio.
Prima di passare alla definizione di RDFS e OWL è necessario introdurre brevemente il concetto di *ontologia*.
La parola *ontologia* ha molteplici definizioni in quanto si presta a diverse interpretazioni in base al contesto, che spazia dall'ambito filosofico a quello informatico; vengono ora riportate quelle più significative.
"In ambito informativo per ontologie intendiamo insiemi di modelli concettuali (definiti anche strutture di metadati) costituiti da accordi sui concetti e sulle relazioni tra di essi, rappresentativi di un dominio di conoscenza"[CAN05].
Sempre nell'ambito dell'informatica, una caratterizzazione più definita descrive la parola ontologia come un sostantivo numerabile con l'iniziale minuscola che si riferisce ad un tipo particolare di information object o di artefatto computazionale, che permette di modellare formalmente la struttura di un certo sistema.
Due definizioni più sintetiche potrebbero essere "An explicit specification of a conceptualisation" [GRUB93] e "A formal specification of a shared conceptualisation" [BOR97].
Per meglio comprendere cosa si intenda per *ontologia* può essere utile descriverne le componenti principali:

- **Classi** (anche concetti o tipi): sono dei predicati unari ognuno dei quali identifica un'entità ontologica che può avere istanze (individui). Esempio: *Studente*

- **Individui**: ciò che viene descritto dall'ontologia, possono essere oggetti concreti (persone, macchine, etc..) o astratti (concetti, funzioni, etc...). Esempio: *Raffaele Giannella*

- **Relazioni**: sono dei predicati binari che riguardano coppie di classi o di individui. Esempio: Raffaele Giannella *è istanza di* Studente

RDFS è un linguaggio che permette di definire il significato, le caratteristiche, le relazioni di un insieme di proprietà, compresi eventuali vincoli sul dominio e sui valori delle singole proprietà. Rende inoltre possibile l'implementazione del concetto di classe e sottoclasse, consentendo di definire gerarchie di classi, con il vantaggio di rendere possibile per agenti software utilizzare queste relazioni per svolgere i loro compiti [SIG02]. RDFS è un'estensione semantica di RDF; fornisce i meccanismi per descrivere gruppi di risorse collegate e le relazioni tra queste risorse [W3C14]. In particolare RDFS permette di creare classi, istanze di classi, sottoclassi, proprietà, sotto-proprietà di altre proprietà e consente inoltre di definire dominio e codominio delle proprietà. Ma RDFS presenta alcune limitazione che hanno portato allo sviluppo di una sua estensione: Web Ontology Language 2 (OWL 2). Con esso si aggiungono molti nuovi costrutti ontologici oltre a quelli definiti per RDFS, permettendo così di aumentare l'espressività di ciò che può essere rappresentato per mezzo di un'ontologia.

2.1.2 Interrogazione: SPARQL

SPARQL è un linguaggio standard W3C[4] per l'interrogazione di repository RDF, rappresenta per RDF quello che SQL è per le basi di dati.
Una query SPARQL si basa sul riconoscimento di pattern "a tripla" su un

[4]http://www.w3.org/TR/sparql11-overview/

grafo RDF, è un meccanismo di "pattern matching": si stabiliscono dei pattern a tripla del tipo "Soggetto-Predicato-Oggetto" simili alle asserzioni RDF, ma nel quale ogni componente è una variabile. Questo meccanismo permette di ricercare corrispondenze all'interno di un grafo RDF. La sintassi è molto simile a quella di SQL, le query hanno infatti una struttura "SELECT-FROM-WHERE", in cui la clausola "FROM" è opzionale se si intende interrogare l'intera base di conoscenza di un certo sistema. Viene fornita inoltre la possibilità di utilizzare la parola chiave "PREFIX" per introdurre dei prefissi assegnati ad un namespace e utilizzabili negli URI delle risorse.

- **SELECT** fornisce l'elenco delle variabili da riportare nel risultato
- **FROM** indica la sorgente dati da interrogare, se omesso si suppone si voglia interrogare l'intera base di conoscenza del sistema
- **WHERE** definisce una serie vincoli sulle triple da estrarre

Questo è un esempio di una semplice query SPARQL:

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { <http://example.org/book/book1>  dc:title  ?title }
```

Dopo aver fissato un prefisso "dc:" il risultato della query sarà il valore della variabile "?title" la quale è l'oggetto nella tripla che ha come soggetto la risorsa "http://example.org/book/book1" e come predicato "dc:title". Sono disponibili anche altri costrutti che permettono di strutturare query anche molto complesse:

- **FILTER**: condizione di filtro utilizzata per specificare un filtro al gruppo di pattern a cui fa riferimento
- **OPTIONAL**: per definire dei pattern opzionali
- **UNION**: per unire i risultati di differenti gruppi di pattern

- **DISTINCT**: precede "SELECT" e serve ad eliminare i duplicati dai risultati della query
- **ORDER BY [ASC, DESC]**: ordina i risultati in base ad una variabile
- **OFFSET**: fornendo un intero, restituisce la lista dei risultati a partire dalla posizione n+1
- **LIMIT**: permette di limitare la dimensione della lista dei risultati

2.1.3 Linked Data e Linking Open Data Project

Il termine *Linked Data* si riferisce ad un insieme di *best practices* per la pubblicazione e la connessione di dati strutturati sul Web: l'adozione di queste *best practices* da parte di un crescente numero di Data Providers ha portato alla creazione di uno spazio globale di dati contenente miliardi di asserzioni: il *Web of Data*[BHB09]. In definitiva, Linked Data riguarda l'utilizzo del Web per creare link tipati tra dati provenienti da sorgenti diverse: questi potrebbero essere dei databases gestiti da organizzazioni diverse o semplicemente sistemi informativi eterogenei che non riescono a comunicare tra di loro perchè adottano standard diversi. I dati devono essere pubblicati sul web in modo tale che siano *machine-readable*, il loro significato deve essere esplicitamente definito, devono essere collegati ad altri dataset esterni e, viceversa, altri dataset devono potersi collegare a questi dati.
Mentre la principale unità di *hypertext* nel web è costituita da documenti HTML collegati per mezzo di *hyperlinks* non tipati, Linked Data si basa su documenti contenenti dati in formato RDF; questi documenti non sono semplicemente collegati tra di loro, ma, per mezzo di RDF vengono fatti degli statements tipati che permettono di collegare arbitrariamente qualsiasi cosa nel mondo. Il risultato, a cui ci si riferisce come "Web of Data", può essere descritto più accuratamente come "web of things in the world, described by data on the Web" [BHB09]. Tim Berners-Lee (2006) ha delineato 4 fon-

damentali principi al fine di pubblicare dati sul Web in modo tale che essi possano diventare parte di un singolo spazio di dati globale[BER06]:

1. Usare URI come nomi per le risorse
2. Usare HTTP URI cosicché le persone possano richiedere le risorse attraverso il loro nome
3. Quando qualcuno richiede una risorsa per mezzo di un URI, devono essere fornite informazioni utili usando gli standard riconosciuti (RDF, SPARQL)
4. Includere link ad altri URI, per rendere possibile la scoperta di nuove cose

Questi 4 principi sono diventati famosi come "*Linked Data principles*".
Il *Linking Open Data Project* è il principale esempio di adozione e applicazione di questi *Linked Data principles*: si tratta di un progetto del W3C[5] nato nel gennaio 2007 il cui obiettivo è quello di estendere il web tradizionale pubblicando dataset RDF liberi e aperti e mettendo in relazione tra loro i dati provenienti da sorgenti diverse. I partecipanti, nelle prime fasi del progetto, erano prevalentemente ricercatori e sviluppatori di laboratori universitari o piccole compagnie. Ma con il passare del tempo il progetto è cresciuto esponenzialmente, grazie soprattutto all'interessamento da parte di grandi aziende che hanno cominciato a pubblicare dati seguendo i principi del Linked Data e hanno collegato tra di loro i datasets esistenti[BHB09].

2.2 Un grande contributo con il Semantic Publishing

Un esempio di applicazione dei principi del Linked Data è rappresentato dal cosiddetto *Semantic Publishing*: l'utilizzo delle tecnologie del Semantic

[5]http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

Web per andare ad arricchire le informazioni relative ad articoli scientifici disponibili online, con l'obiettivo di migliorare la comunicazione all'interno della comunità scientifica. Il mondo dell'editoria scientifica è stato rivoluzionato negli ultimi 10 anni grazie all'avvento e alla diffusione delle tecnologie digitali che hanno reso possibile pubblicare articoli scientifici online, accessibili facilmente dai moltissime persone, soprattutto ricercatori legati alla comunità scientifica. Ma nonostante questa rivoluzione digitale, la struttura fondamentale degli articoli scientifici è rimasta sostanzialmente inalterata e il mondo dell'editoria ancora non sfrutta appieno le potenzialità offerte dal World Wide Web[SHO09]; in altre parole, la fruizione di questi contenuti digitali è rimasta, in molti casi, ancora legata al passato e non si fa uso delle tecnologie che oggigiorno il web mette a disposizione: gli articoli scientifici vengono presentati come una semplice pagina HTML o un PDF statico, il contenuto c'è tutto, ma non c'è interattività, l'utente non ha accesso, ad esempio, alle informazioni relative agli articoli citati o ad altri articoli ritenuti rilevanti. Il semantic publishing rappresenta un passo avanti nella direzione di un'evoluzione del mondo dell'editoria scientifica, con l'intenzione di realizzare nuove forme di fruizione dei contenuti digitali al fine di rendere più efficiente la produzione e la condivisione della conoscenza scientifica.

2.2.1 Cos'è il Semantic Publishing

Il Semantic Publishing riguarda l'uso del Web e delle tecnologie del Semantic Web per arricchire un documento pubblicato, ad esempio un journal article, in modo da rendere possibile la definizione di una rappresentazione formale del suo significato, facilitare la scoperta automatica, permettere il collegamento con altri articoli semanticamente correlati, fornire l'accesso ai dati all'interno dell'articolo in modo interattivo e consentire l'integrazione di dati tra diversi articoli.[SHO09][PER14].
Questo arricchimento consiste nell'aggiunta di opportuni metadati che siano facilmente trattabili attraverso elaborazioni e analisi automatizzate, consentendo quindi una maggiore verificabilità delle informazioni pubblicate.

Questi arricchimenti semantici incrementano il valore intrinseco di un articolo perchè consentono di estrarre informazioni, comprensione e conoscenza in modo più semplice[SHO09]. Con il semantic publishing si rende inoltre possibile lo sviluppo di servizi evoluti che permettono ai lettori di accedere in modo rapido ed efficiente a queste informazioni strutturate.

Prima di poter realizzare tali servizi è innanzitutto necessario rendere disponibili i dati strutturati; negli ultimi anni sono stati pubblicati diversi dataset come LOD (Linked Open Data) con l'obiettivo di fornire informazioni strutturate su articoli scientifici; l'attuale panorama è molto variegato e molte informazioni sono disponibili attraverso diversi SPARQL endpoint[BCD14]. Sono presenti diversi sistemi che supportano l'esplorazione di dati relativi ad articoli scientifici e rendono tali dati accessibili come Linked Open Data
Alcuni esempi sono:

- Nature Publishing Group LOD Platform[6] si concentra sui dati bibliografici
- DBLP++[7] si focalizza sui dati relativi alla authorship e non contiene alcune informazione relative alle citazioni
- JISC OpenCitation corpus[SHO13] fornisce liberamente una grande quantità di dati sugli articoli pubblicati in Open Access PubMed Central[8]; questi dati riguardano le citazioni, gli abstract, gli autori, ma non sono disponibili per tutti gli articoli
- BioTea project[GMC13] si occupa di rendere la letteratura biomedica disponibile in formato RDF, prendendo gli articoli da PubMed Central

Per l'applicazione sviluppata per questo progetto di tesi si è lavorato su un altro dataset, si tratta del Semantic Lancet Triplestore realizzato per il Semantic Lancet Project; nella sezione seguente verranno descritti i tratti fondamentali di questo progetto.

[6] http://www.nature.com/developers/documentation/linked-data-platform/
[7] http://dblp.l3s.de/dblp++.php
[8] http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

2.3 Semantic Lancet Project

Il Semantic Lancet Project[9] è incentrato sulla realizzazione di un Linked Open Dataset su pubblicazioni scientifiche [BCD14]; l'obiettivo del progetto è duplice:

- produrre dati RDF basati sull'utilizzo delle *Semantic Publishing and Referencing* (SPAR) *Ontologies*[10]
- rendere pubblicamente accessibile il triplestore RDF[11] tramite SPARQL endpoint e fornendo una serie di servizi costruiti a partire dai dati sugli articoli del *Journal of Web Semantics*[12] forniti da *Elsevier*[13]

Il framework di Semantic Lancet Project è suddiviso in tre macro-sezioni:

1. **data reengineering**: responsabile della conversione di *raw data*, provenienti da repository esterne (i.e. Scopus[14] e ScienceDirect[15]), in OWL conformemente alle ontologie SPAR; questi dati vengono poi integrati nel triplestore
2. **semantic enhancement**: partendo dai dati nel Semantic Lancet Triplestore, essi vengono arricchiti semanticamente con informazioni provenienti da diverse fonti
3. **services**: vengono fornite applicazioni per consentire alle persone di navigare ed esplorare i dati del SLT in modo semplice ed intuitivo, nascondendo cioè la complessità dei dati e delle tecnologie sottostanti, fornendo agli utenti un vista ad alto livello sul contenuto del dataset; L'applicazione sviluppata, BEX, va ad arricchire l'insieme di questi strumenti

[9] http://www.semanticlancet.eu/
[10] http://www.sparontologies.net
[11] il triplestore attualmente utilizzato è Fuseki
[12] http://www.journals.elsevier.com/journal-of-web-semantics/
[13] http://www.elsevier.com/
[14] http://www.scopus.com
[15] http://sciencedirect.com

Prima di passare alla descrizione del Semantic Lancet Triplestore è necessario spiegare cosa sia il modello FRBR e illustrare le ontologie che stanno alla base della struttura del dataset.

2.3.1 Livelli FRBR

Il Functional Requirements for Bibliographic Record (FRBR)[IFL09] è un modello proposto dall'International Federation of Library Association (IFLA)[16] con l'intenzione di delineare, per mezzo di termini chiaramente definiti, le funzioni svolte da un record bibliografico rispetto ai vari media, alle varie applicazioni e ai vari bisogni dell'utente. FRBR presenta un modello di record essenziale ed esauriente rispetto alle esigenze e alle aspettative del lettore[IFL09].
Il modello prende in considerazione l'intero spettro delle funzioni di un record bibliografico nella sua accezione più ampia, ossia delle funzioni che riguardano gli elementi descrittivi e, in prospettiva, i punti di accesso. FRBR è un modello con il quale si cerca di stabilire cosa debba fornire un record bibliografico e quale sia il suo fine, in un'ottica di catalogazione e di garanzia di qualità.
Questo modello stabilisce quattro livelli diversi di informazioni relative ad un record:

1. **Opera (Work)**: una opera intellettuale o artistica. Si tratta di una entità astratta che non rappresenta nessun oggetto fisico specifico. Esempio: l'Iliade di Omero

2. **Espressione (Expression)**: è la realizzazione di un work sotto forma di testo, immagini, suoni o una loro combinazione. Esempio: un'espressione scritta dell'Iliade o un'espressione sotto forma di opera teatrale

3. **Manifestazione (Manifestation)**: definisce la rappresentazione fisica di una specifica espressione. Esempio: una specifica edizione scritta dell'Iliade

[16] http://www.ifla.org/

4. **Unità (item)**: è il singolo esemplare di una manifestazione. Esempio: la singola copia di una specifica edizione scritta dell'Iliade

2.3.2 SPAR: Semantic Publishing And Referencing Ontologies

SPAR[17] è una suite di ontologie OWL 2 il cui obiettivo è quello di descrivere gli aspetti rilevanti del processo di pubblicazione, con particolare attenzione alle pubblicazioni scientifiche[DGP15]; esse rappresentano le principali ontologie utilizzate per la descrizione dei dati del Semantic Lancet Triplestore. Qui di seguito viene data una rapida descrizione di queste ontologie:

- **FaBiO**: FRBR-aligned Bibliographic Ontology; descrive le entità che sono state pubblicate o che sono potenzialmente pubblicabili e che contengono riferimenti bibliografici o sono menzionate da essi. Le classi di FaBiO sono strutturate sul modello dei livelli FRBR: Works, Expressions, Manifestations e Items. Questa ontologia sostanzialmente descrive i metadati fondamentali associati ad un articolo (autori, titolo, DOI, etc...)

- **CiTO**: Citation Typing Ontology; permette di caratterizzare la natura o il tipo delle citazioni. Ad esempio la proprietà *cito:cites* permette di indicare che un articolo ne cita un altro, ci sono inoltre un ampio insieme di sotto-proprietà di *cito:cites* che permettono di caratterizzare la tipologia di citazione, il motivo della citazione: *citation functions*(es. *cito:extends*, *cito:usesMethodIn*, etc...)

- **BiRO**: Bibliographic Reference Ontology; permette di definire i record bibliografici, i riferimenti bibliografici e la loro compilazione in collezioni e liste

- **C4O**: Citation Counting and Context Characterization Ontology; permette di registrare il numero di citazioni *in-text*(nel testo di un artico-

[17]http://sempublishing.sourceforge.net/

lo) ricevute da un certo articolo citato, oltre al suo numero di citazioni globali. Essa permette inoltre di caratterizzare il contesto citazionale (*citation context*) di ogni citazione

- **DoCO**: Document Components Ontology; ontologia per la caratterizzazione delle varie parti di un documento, sia strutturali (es. sezione, paragrafo), che retoriche (es. introduzione, discussione, etc...)
- **PRO**: Publishing Roles Ontology; permette di specificare i ruoli di agenti (autori, editori, etc...) coinvolti nel processo di pubblicazione.
- **PSO**: Publishing Status Ontology; caratterizza lo stato della pubblicazione di un documento in ogni fase del processo di pubblicazione (es. in fase di review, pubblicato, etc...)
- **PWO**: Publishing Workflow Ontology; descrive i passi del flusso di lavoro associato alla pubblicazione di un documento o di altre tipologie di pubblicazioni

L'utilizzo di queste ontologie permette di definire elementi bibliografici, record bibliografici e riferimenti, citazioni, numero di citazioni, contesti citazionali e le loro relazioni con sezioni rilevanti degli articoli citati. E ancora, l'organizzazione di record bibliografici, liste ordinate di riferimenti, sezioni dei singoli articoli, ruoli nel processo di pubblicazione e lo stato di pubblicazione degli articoli.

2.3.3 Il triplestore

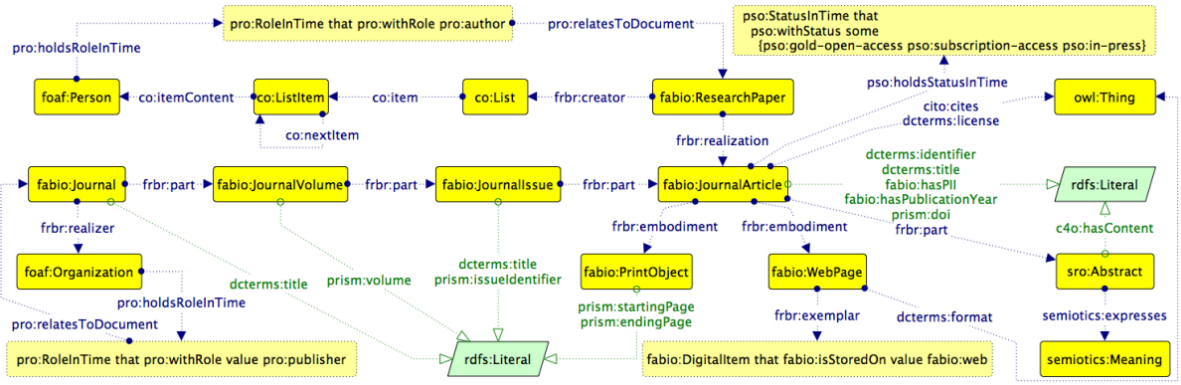

Figura 2.1: Data model dei Journal Articles in SLT

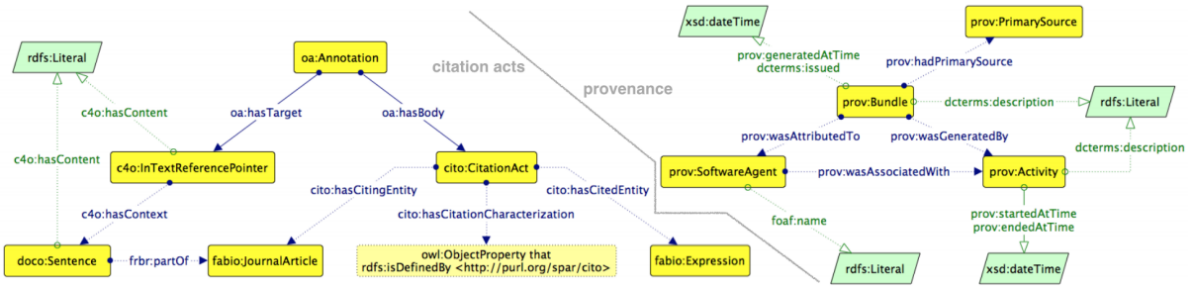

Figura 2.2: Data model per i Citation Acts

Il Semantic Lancet Triplestore è un LOD dataset liberamente accessibile e progettato con l'intenzione di fornire una grande quantità di dati su articoli scientifici; oltre alle informazioni bibliografiche di base e ai dati di authorship vengono resi disponibili dettagli relativi agli abstract, alle citazioni, ai motivi citazionali, ai contesti citazionali e in generale alla rete citazionale degli articoli scientifici; quest'ultima permette di collegare un articolo ad un insieme di altri articoli ritenuti rilevanti ed in qualche modo affini a quanto trattato nell'articolo. Tali informazioni sono disponibili grazie all'arricchimento semantico di dati estratti direttamente dal contenuto testuale degli articoli. L'intenzione è quella di fornire una visione più ampia ed approfondita sulle

relazioni che legano papers scientifici rendendo inoltre il loro contenuto esplicitamente formalizzato.

In Figura 2.1 è illustrato il modello dei dati utilizzato nel dataset per la rappresentazione dei *journal articles* e nella parte sinistra della Figura 2.2 è presente il modello che permette di descrivere gli atti citazionali (*citation acts*) all'interno degli articoli, i motivi citazionali (*citation functions*) e le frasi nelle quali si parla di un articolo citato (*citation context*); oltre alle ontologie SPAR, vengono utilizzate anche altre ontologie, come DCTerms[18], PRISM[19], Collections Ontology (CO)[20], Open Annotation Data Model[21] e frbr[22]. In questo contesto, per comprendere meglio come siano strutturate le informazioni all'interno del dataset, può essere utile fare degli esempi mostrando properties e values delle risorse relative ad un articolo; verrà usato un *resource browser* per visualizzare tali informazioni. In Figura 2.3 sono rappresentate le risorse accessibili al livello frbr *Work* di un articolo dal titolo "A general Datalog-based framework for tractable query answering over ontologies"; da qui si possono ottenere le informazioni sulla tipologia di articolo (*research paper* nell'esempio), sugli autori dell'articolo (property *creator*) o ulteriori dettagli sull'articolo spostandosi al livello *Expression*, cioè la realizzazione del *work*.

Al livello *Expression* (Figura 2.4) vengono fornite altre informazioni essenziali, ad esempio l'abstract, l'anno di pubblicazione, il DOI, il tipo di articolo, le citazioni globali e i già menzionati *cites* che puntano ad altre risorse di livello *Expression* le quali rappresentano gli articoli citati dall'articolo analizzato; è possibile inoltre, per mezzo della property *embodiment*, "scendere" ancora al livello di *manifestation* per avere informazioni, come nell'esempio in Figura 2.5, sulla manifestazione dell'articolo in formato di pagina web html (è disponibile un link ipertestuale come valore della property *exemplar*).

[18] http://dublincore.org/documents/dcmi-terms/
[19] http://www.prismstandard.org/resources/mod_prism.html
[20] https://code.google.com/p/collections-ontology/
[21] http://www.openannotation.org/spec/core/
[22] http://vocab.org/frbr/core.html

Property	Value
label	*'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Work)*
type	work
type	research paper
realization	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Expression)
creator	List of the authors of 'A general Datalog-based framework for tractable query answering over ontologies'

Figura 2.3: esempio Work

Per avere informazioni sul tipo di citazione, la risorsa fondamentale è costituita dal *Citation Act* (Figura 2.6) il quale collega una *citing entity* (articolo citante) ad una *cited entity* (articolo citato) e fornisce il dettaglio sulla caratterizzazione dell'atto citazionale (*citation characterization*). Da sottolineare il fatto che ci possano essere più *citation acts* con *citing entity* un articolo "X" e *cited entity* un articolo "Y", ma al livello *Expression* dell'articolo "X" ci sarà una sola property *cites* con value l'*Expression* dell'articolo "Y". La Figura 2.7 presenta un esempio di *In Text Reference Pointer* il quale identifica la posizione, nel testo di un articolo, nella quale è presente un puntatore ad un elemento della bibliografia; la property *has context* ha come valore una *sentence* (Figura 2.8 che permette di accedere al contenuto testuale (*has content*) presso il quale viene effettuata la citazione.

Property	Value
part	Abstract of 'A general Datalog-based framework for tractable query answering over ontologies'
label	*'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Expression)*
type	expression
type	journal article
embodiment	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Printed Manifestation)
embodiment	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR HTML Manifestation)
identifier	*1-s2.0-S1570826812000388*
realization of	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Work)
holds status in time	Access status referring to 'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Expression)
has p i i	*S1570-8268(12)00038-8*
doi	*10.1016/j.websem.2012.03.001*
title	*A general Datalog-based framework for tractable query answering over ontologies*
has publication year	*2012*
cites	'Query Answering for OWL-DL with rules' (FRBR Expression)
cites	'On the decidability and complexity of integrating ontologies and rules' (FRBR Expression)
cites	'OWL rules: A proposal and prototype implementation' (FRBR Expression)
cites	'A comparison of two modelling paradigms in the Semantic Web' (FRBR Expression)
has global citation frequency	Scopus cited-by count: 30 (dated 2014-05-02)

Figura 2.4: esempio Expression

Property	Value
label	*'A general Datalog-based framework for tractable query answering over ontologies' (FRBR HTML Manifestation)*
type	manifestation
type	web page
format	html
embodiment of	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Expression)
exemplar	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR HTML Item)

Figura 2.5: esempio Manifestation

Property	Value
type	citation act
has citing entity	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Expression)
has cited entity	'On the decidability and complexity of integrating ontologies and rules' (FRBR Expression)
has citation characterization	confirms

Figura 2.6: esempio Citation Act

Property	Value
type	in text reference pointer
has content	*[11]*
denotes	reference
has context	sentence with in text reference pointer 45

Figura 2.7: esempio In Text Reference Pointer

Property	Value
type	sentence
part of	'A general Datalog-based framework for tractable query answering over ontologies' (FRBR Expression)
has content	*Note that EGDs generalize functional dependencies (FDs) and, in particular, key dependencies (or keys) [11].*

Figura 2.8: esempio Sentence

2.3.4 Necessità di servizi e interfacce user-friendly per esplorare il dataset

Il Semantic Lancet Triplestore, in definitiva, mette a disposizione una grande quantità di informazioni strutturate, ma, per poter considerare completo il framework di questo progetto, è necessario fornire strumenti e servizi che permettano di esplorare il dataset e trarre informazioni utili dai dati in esso contenuti. Il triplestore è interrogabile tramite SPARQL endpoint, ma questa modalità di fruizione dei contenuti può essere considerata come un'interfaccia utilizzabile solo dai "tecnici" in quando richiede un'approfondita conoscenza delle tecnologie del semantic web e della struttura dei dati interni al dataset. Anche il *resource browser* utilizzato per gli esempi della sezione precedente non può essere considerato come uno strumento di navigazione utilizzabile dall'utente medio il quale non ha alcuna conoscenza delle ontologie alla base del modello dei dati. Si rende dunque necessario costruire dei servizi che garantiscano un alto livello di astrazione e rendano totalmente trasparente (agli utenti) la complessità delle tecnologie e dei dati sottostanti.
Un esempio di questo tipo di servizi è il *Data Browser*[23] che mette a disposizione un'interfaccia user-friendly per mostrare le informazioni sugli articoli di un certo autore: vengono fornite le informazioni bibliografiche generiche, la lista degli autori dell'articolo, l'abstract e il numero di citazioni globali ricevute dal singolo articolo. Questo strumento può essere utile, ad esempio, per valutare la produttività di un determinato ricercatore.
Un altro servizio è l'*Abstract Finder*[24] il cui obiettivo principale è quello di supportare un ricercatore nelle attività di ricerca di articoli che trattino una determinata tematica; l'utente effettua una ricerca testuale e il servizio si occupa di recuperare le informazioni su una lista di articoli che nel loro abstract trattano di concetti riconducibili al testo inserito dall'utente.
L'intenzione di BEX è quella di integrare le funzionalità degli strumenti descritti in precedenza con l'aggiunta di nuove funzionalità che permettano di

[23] http://www.semanticlancet.eu/browser
[24] http://www.semanticlancet.eu/abstractfinder

fornire all'utente ulteriori informazioni utili e consentano una navigazione attraverso la rete citazionale degli articoli.

Capitolo 3

BEX: una web app task-oriented di supporto ai ricercatori

In questo capitolo vengono inizialmente introdotti i principali *tasks* presi in considerazione, i principi, le idee e i modelli che hanno guidato la progettazione dell'interfaccia utente. In seguito verranno descritte le funzionalità messe a disposizione dall'applicazione.

3.1 L'idea di base dell'applicazione

Cos'è BEX? è un'applicazione web task-oriented il cui principale obiettivo è quello di supportare i ricercatori universitari nell'esecuzione di determinate attività di ricerca e valutazione che, per la loro esecuzione, prevedano l'analisi di grandi quantitativi di informazioni riguardanti articoli scientifici; ciò si ottiene fornendo agli utenti delle interfacce user-friendly e delle funzionalità (di ricerca, navigazione e filtering) appositamente progettate affinché essi possano portare a compimento i loro tasks in modo semplice ed efficiente senza doversi confrontare con gli inconvenienti che spesso si manifestano con l'uso di una digital library *general purpose* (sezione 1.3.2). Come detto in

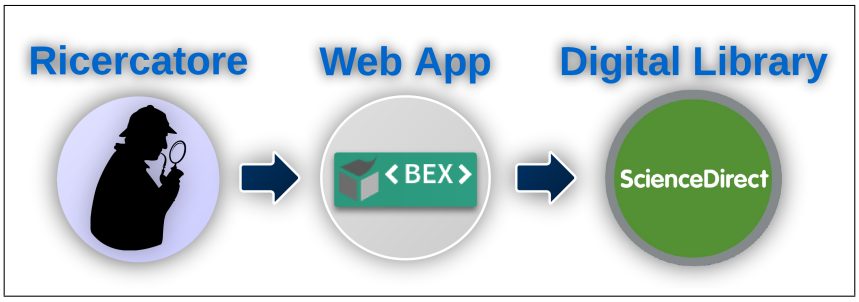

Figura 3.1: Bex come mediatore tra il ricercatore e la DL

precedenza, BEX si configura come una sorta di livello intermedio (vedere figura 3.1) tra il ricercatore e la digital library che detiene materialmente i contenuti digitali: il ricercatore utilizza l'applicazione per effettuare le ricerche, navigare tra i risultati, effettuare le valutazioni considerate importanti per poi passare, se ritenuto necessario, alla digital library per ottenere l'accesso al contenuto digitale vero e proprio.
Questa efficienza e semplicità d'uso facilita il raggiungimento degli obiettivi prefissati (goals) dal ricercatore in quanto permette di risparmiare in termini di tempo di esecuzione del task e di risorse cognitive, favorendo così l'attività di sensemaking.
Lo sviluppo di questa applicazione è contestuale al *Semantic Lancet Project*: il *Semantic Lancet Triplestore* mette a disposizione una grande quantità di informazioni (dati e metadati) relative ad articoli scientifici[1] e alla loro rete citazionale, ma queste informazioni, come evidenziato nel capitolo precedente, erano (prima dello sviluppo di BEX) difficilmente accessibili agli utenti poichè gli strumenti messi a loro disposizione erano difficilmente utilizzabili dall'utente medio senza conoscenze approfondite delle tecnologie alla base del semantic web o le funzionalità che venivano esposte da determinati servizi non permettevano di sfruttare appieno le potenzialità del dataset sottostante.

[1] allo stato attuale, il dataset descrive tutti gli articoli pubblicati in "Journal of Web Semantics" (Elsevier): http://www.journals.elsevier.com/journal-of-web-semantics

3.2 I tasks

Come detto in precedenza, questa applicazione ha come target uno specifico segmento di utenza, cioè i ricercatori universitari, i quali in base al ruolo che si trovano a ricoprire, effettuano determinati tasks. Per chiarezza espositiva è utile specificare due definizioni riportate in [PRS94]:

- **Goal**: l'obiettivo finale che l'utente vuole raggiungere
- **Task**: la sequenza di una o più attività che l'utente ritiene necessarie per raggiungere un goal

Si ritiene necessario sottolineare che i goals e tasks che verranno esposti e descritti non sono rappresentativi di tutti i possibili obiettivi che un ricercatore potrebbe avere in mente; l'applicazione è stata progettata e sviluppata con la precisa intenzione di facilitare l'esecuzione di un relativamente ristretto insieme di tasks: ciò contribuisce a rendere BEX non dispersiva o complicata all'uso.
In figura 3.2 vengono descritti brevemente i goals che il ricercatore vuole raggiungere, i principali tasks che ha in mente per farlo e il ruolo che si trova a ricoprire; queste osservazioni sono frutto di un confronto diretto con un gruppo di ricercatori e sono in parte ispirate a quanto riportato in [DGP15].
Un task fondamentale è sicuramente il ***task 1*** in cui il ricercatore, nel ruolo di lettore o scrittore, ha la necessità di redigere (o aggiornare) una lista di articoli che trattano una certa tematica. Ciò è necessario, ad esempio, quando il ricercatore come autore si trova a definire la sezione "*related works*" di un paper che sta scrivendo o quando egli vuole mantenersi aggiornato su una certa area di ricerca. Il ricercatore parte da una ricerca generica basata su alcune parole chiave, scorre la lista dei risultati ed effettua una prima selezione di quelli ritenuti interessanti; di questi ne approfondisce l'analisi leggendone l'abstract, analizzandone gli elementi della bibliografia e le citazioni ricevute. In seguito, molto spesso il ricercatore è interessato ad alcuni elementi della bibliografia o ad articoli citanti: questi sono nuovi candidati

da valutare e l'utente ne approfondisce lo studio. Questo processo potrebbe essere ripetuto anche più volte, permettendo così al ricercatore di costruire la sua lista di articoli interessanti in modo incrementale. Sarebbe quindi molto utile potersi muovere facilmente attraverso la rete citazionale (articoli citati e citanti) di un articolo. Questo tipo di esplorazione richiede che l'utente elabori una grande quantità di informazioni relative a molti articoli e spesso questa operazione è eseguita ripetutamente e sarebbe conveniente non impiegarci molto tempo.

Un altro task molto importante è il ***task 2***: il ricercatore nel ruolo di recensore deve esprimere dei giudizi su un certo articolo, valutandolo sotto diversi punti di vista. Con "valutare la freschezza e la rilevanza" si intende stabilire quanto sia "attuale" l'argomento trattato nell'articolo, il ricercatore vuole capire se l'articolo si occupa di tematiche su cui c'è un certo fervore da parte della comunità scientifica, se gli autori dell'articolo stanno lavorando molto su un certo progetto; egli si pone domande come "quando è stato pubblicato questo articolo?", "gli articoli citati (dall'articolo preso in considerazione) sono recenti? che tipo di articoli cita? per quale motivo li cita?", "come ha reagito la comunità scientifica a questo articolo?" e ancora "ci sono delle autocitazioni da/verso articoli che trattano lo stesso argomento? gli autori stanno ancora portando avanti il progetto trattato in questo articolo?"; anche nel caso di questo task risulta funzionale un'analisi degli elementi della bibliografia e degli articoli citanti con particolare attenzione verso gli anni di pubblicazione, citazioni e motivi citazionali (e contesti citazionali).

Come già sottolineato, il ricercatore potrebbe essere interessato nell'indagine delle cosiddette autocitazioni di un articolo, cioè citazioni verso o da articoli che con cui l'articolo esaminato ha in comune almeno un autore; la presenza di autocitazioni permette di fare alcune osservazioni, ad esempio, su come un certo gruppo di ricercatori stia lavorando su un certo progetto, di come esso si sia evoluto nel tempo. Questo è l'obiettivo del ***task 3***, valutare le autocitazioni di uno o più articoli e per poterlo fare è necessario controllare gli autori di tutti gli articoli citati, ma potrebbe essere utile considerare anche

gli articoli citanti.
Il ***task 4*** prevede attività simili ai precedenti tasks ma in questo caso l'attenzione è posta più sull'evoluzione nel tempo delle reazioni della comunità scientifica verso un certo articolo: il ricercatore ha interesse nello studiare come siano variate citazioni e motivi citazionali verso l'articolo, anno per anno, al fine di quantificare e qualificare l'impatto che quell'articolo ha determinato. ***task 5*** e ***task 6*** vedono il ricercatore nel ruolo di "workshop chair" che ha come obiettivo quello di cercare dei possibili partecipanti ad una conferenza o nel ruolo di lettore/autore in cerca di potenziali collaboratori per un progetto; la ricerca parte dall'analisi di un insieme di articoli rilevanti (e recenti) che trattano un certo argomento per poi passare ad esaminare gli autori di questi articoli per valutare quali di questi possano essere considerati degli esperti. Anche in questo caso può risultare conveniente muoversi attraverso la rete citazionale degli articoli.
I tasks fin qui analizzati sono da considerarsi casi generali, si potrebbero infatti fare esempi di tasks maggiormente dettagliati che articolino le attività descritte in modo diverso, ma sarebbero comunque riconducibili in qualche modo ad uno dei casi trattati; per esigenze di semplicità espositiva questi casi non verranno descritti.
Si può osservare come in tutti i tasks sia più o meno costante lo studio (oltre alle informazioni generiche) di alcuni elementi in particolare, relativi agli articoli oggetto di approfondimento: la bibliografia (articoli citati) e le citazioni (articoli citanti). Queste componenti informative oltre a consentire considerazioni più definite su un articolo, rappresentano il ponte di collegamento tra questo articolo e gli altri articoli che gli gravitano attorno e permettono al ricercatore di muoversi all'interno della rete citazionale di articoli scientifici riguardanti una certa area di ricerca di interesse per l'utente: per gli scopi dell'applicazione questo è sicuramente il modello migliore di navigazione da fornire all'utente. Ulteriori osservazioni sui tasks sono proposte nella successiva sezione.

#	Ruolo	Goal	Task
1	lettore/autore	realizzare/aggiornare una bibliografia di articoli (recenti) su un certo tema	• trovare articoli (recenti) che parlano del tema nel loro abstract • controllare le citazioni (recenti) ricevute • controllare gli articoli citati • navigare la rete citazionale attraverso gli elementi della bibliografia e gli articoli citanti
2	recensore	valutare la freschezza e la rilevanza di un articolo	• controllare l'anno di pubblicazione dell'articolo • controllare l'anno di pubblicazione degli articoli citati • controllare motivi e contesti citazionali per gli articoli citati • controllare motivi e contesti citazionali per gli articoli che lo citano
3	recensore/editor	valutare le autocitazioni di un articolo	• controllare gli autori di tutti gli articoli citati
4	valutatore	valutare l'impatto di un articolo	• controllare quante volte l'articolo è stato citato • controllare i motivi per cui viene citato (e i contesti citazionali) • controllare come si sono evolute le citazioni nel tempo
5	workshop chair	cercare potenziali partecipanti	• controllare gli autori di articoli rilevanti (e recenti) riguardo un certo tema • analizzare gli altri articoli di questi autori
6	lettore/autore	cercare collaboratori per un progetto	• controllare gli autori di articoli rilevanti (e recenti) riguardo un certo tema • analizzare gli altri articoli di questi autori

Figura 3.2: Goals e tasks

3.3 Note su User Interface Design, interazione e User Experience

Prima di passare ad una descrizione dettagliata delle singole funzionalità fornite dall'applicazione, si ritiene utile esporre quali siano le idee, i principi, i modelli e le linee guida generali che hanno orientato l'intera attività di progettazione.
BEX è un'applicazione web che recupera e gestisce informazioni *complesse* e ha come principale obiettivo quello di presentarle all'utente in modo *semplice*: "Complex information can be presented without the user having any knowledge of the complexity" [ABO97].
Questo complicato compito di "trasformazione" dal complesso al semplice spetta all'interfaccia; è al livello dell'interfaccia che avviene l'interazione tra due sistemi complessi: la macchina (computer) e l'utente (persona). La tipologia e la qualità dell'interazione determinano il livello di soddisfazione percepito dall'utente: basta la mancata comprensione di pochi elementi a portare l'utente a sentirsi confuso e disorientato, trasformando così l'interazione da un'esperienza positiva ad una negativa e da dimenticare.
Per affrontare lo studio e l'analisi dell'interazione ci sono diversi modelli che ne forniscono una formalizzazione, essi sono astrazioni che forniscono delle cornici per la comprensione dell'interazione; le seguenti osservazioni si ispirano al modello di Abowd e Beale [ABD98].

3.3.1 Modello di Abowd e Beale

Come descritto dalla figura 3.3, l'interazione è formata da 4 componenti, che usano linguaggi diversi:

- **sistema**: gestisce le informazioni e ne rende possibili il recupero e la presentazione, esso utilizza un suo linguaggio di base, detto *core*
- **input**: utilizza il linguaggio di *input*
- **output**: utilizza il linguaggio di *output*

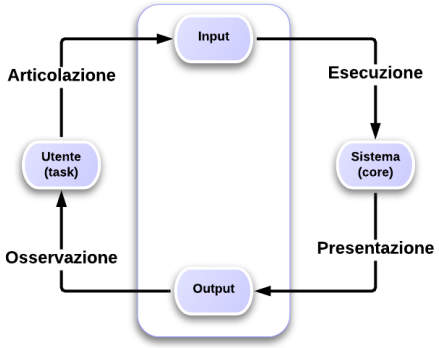

Figura 3.3: Modello di Abowd e Beale

- **utente**: è colui che usa il sistema interattivo e utilizza il linguaggio di *task*

In particolare, dettagliando ulteriormente quanto detto nei capitoli precedenti, l'utente usa il sistema interattivo per raggiungere i suoi goals (obiettivi, ciò che vuole raggiungere), nell'ambito di un certo dominio (un'area applicativa oggetto di studio); per realizzare questi obiettivi l'utente ha in mente dei tasks (intenzioni, sequenze di azioni).
L'interazione può essere interpretata come un'operazione di traduzione tra i vari linguaggi, quindi un sistema interattivo non fa altro che risolvere problemi di traduzione.
Questo è un esempio di ciclo di interazione:

1. l'utente ha un obiettivo
2. l'obiettivo è tradotto in azioni per mezzo dell'input dell'interfaccia (**articolazione**)
3. queste azioni sono tradotte in operazioni sul sistema, il quale cambia di stato (**prestazione**)
4. questo stato è tradotto nell'output dell'interfaccia (**presentazione**)
5. l'output è infine interpretato dall'utente (**osservazione**)

Il compito del progettista di un sistema di interazione (l'applicazione) è quello di trovare traduzioni adeguate tra i vari linguaggi, egli si deve porre domande del tipo "come rendere facile l'articolazione di un task da parte dell'utente?", "l'utente nell'osservare l'output riesce a trovare una correlazione con i propri obiettivi?". Nello sviluppo dell'applicazione, per arrivare all'implementazione di un sistema di interazione che favorisse le attività descritte, si è richiesto il confronto con diverse questioni di design.

3.3.2 Design di dialogo

L'interazione è una sorta di dialogo tra l'utente e la macchina; lo stile dell'interazione ne determina l'efficacia. Lo stile adottato per l'applicazione sviluppata è il cosiddetto "menu e navigazione": l'interazione avviene attraverso la selezione, da parte dell'utente, di una o più opzioni tra quelle visualizzabili in un certo contesto, le opzioni sono organizzate in raggruppamenti con l'intenzione di facilitare le attività dell'utente. Questo stile di interazione permette l'utilizzo dell'applicazione anche da parte degli utenti meno esperti, perlomeno finchè il sistema non è troppo complesso.

3.3.3 Presentazione delle informazioni

"Less is more"

Ludwig Mies van der Rohe

Come presentare le informazioni all'utente? cosa mostrare e cosa nascondere? quando mostrare certe informazioni all'utente? Per rispondere a queste domande, nel corso dello sviluppo dell'applicazione, sono stati tenuti a mente alcuni principi, linee guida, euristiche e consigli proposti nella letteratura del settore, ad esempio si è tratto spunto da molte considerazioni presenti in [GAR10], [HHS15] , da "le 8 regole d'oro dell'*Interface Design*" proposte da Ben Shneiderman [SHP10] e da "le 10 euristiche dell'usabilità di Nielsen e Molich"[NM90].

Minimizzare il carico cognitivo

L'interazione deve richiedere meno risorse cognitive possibili all'utente, cioè limitare l'impatto su memoria sensoriale (iconica in particolare) e a breve termine. La memoria sensoriale è il primo stadio nel processo di memorizzazione e si riferisce ai processi percettivi sensoriali; la memoria iconica, in particolare, si riferisce ai processi percettivi visivi. La permanenza delle informazioni a questo livello è molto breve, pochi istanti e solo poche informazioni vengono mantenute per più tempo; minimizzare l'impatto sulla memoria sensoriale vuol dire ridurre lo "sforzo" richiesto per interpretare lo stimolo: a questo fine è essenziale fornire interfacce che presentino un numero non elevato di elementi, questi devo essere differenziati, cioè ben distinguibili da tutti gli altri e devono occupare una posizione stabile nell'interfaccia. La memoria a breve termine, detta anche memoria di lavoro, è lo stadio successivo nel processo di memorizzazione: essa può mantenere pochi elementi di informazione (circa 7 "*chunks*") per un lasso di tempo stimabile nelle poche decine di secondi (10-20 secondi). Di competenza di questo livello sono la memorizzazione e l'elaborazione delle informazioni legate alle attività che si stanno svolgendo per l'esecuzione di un task; la riduzione dell'impatto sulla memoria di lavoro si ottiene minimizzando il numero di elementi informativi che l'utente deve tenere a mente nel passaggio da una schermata all'altra, organizzando le informazioni mostrate in raggruppamenti e rendendo possibile il completamento di un task attraverso poche azioni e pochi cambi di contesto. Anche alcune delle soluzioni proposte qui di seguito contribuiscono ad abbassare il carico cognitivo.

Coerenza e standard

Le funzionalità offerte dal sistema devono essere coerenti: se l'utente effettua una sequenza di azioni per arrivare ad un risultato, allora quella stessa sequenza di azioni, eseguita in un'altra parte nel sistema, deve portare a risultati simili. Se si utilizzano delle convenzioni grafiche, di terminologia o di formato, allora queste convenzioni devono essere consistenti in tutto il siste-

ma: ad esempio, le icone e i termini utilizzati devono avere sempre lo stesso significato e le stesse connotazioni ovunque essi compaiano, che si tratti di menu, singoli elementi o finestre di dialogo; riassumendo il concetto, se si usano degli standard, questi devono essere consistenti.
Ma il sistema non deve essere coerente solo "con sé stesso", ma è auspicabile che lo sia anche con gli altri sistemi e applicazioni utilizzati dagli utenti, che sia dunque "coerente con l'esterno"; il concetto può essere introdotto dalla *Jakob's Law of the Internet User Experience*: "la gente passa la maggior parte del tempo online su altri siti".
Questo vuol dire che gli utenti preferiscono utilizzare siti che si adattano alle aspettative che si sono costruite nell'utilizzo di altri sistemi. Un utente abituato ad usare un sistema che adotta determinati modelli concettuali e convenzioni, troverà immediatamente familiare un nuovo sistema che, seppur differente, si attiene anch'esso agli stessi modelli e convenzioni; ciò permette di rendere più veloce l'apprendimento dell'uso del nuovo sistema da parte dell'utente.
Un elemento importante è costituito dal vocabolario: utilizzare parole, termini, concetti e metafore che fanno parte del modello culturale proprio dell'utente in modo da permettergli di stabilire facilmente delle correlazioni logiche facilitando così la comprensione delle funzionalità fornite dal sistema. Discorso simile può essere fatto per le icone che devono seguire certi standard e non essere ambigue.
Per riassumere in breve: "differenziare ma non diversificare il sistema". In questo contesto con il termine "diversificare" si intende il fornire all'utente un sistema, un'applicazione che esca totalmente dagli standard preesistenti, che utilizzi convenzioni e modelli concettuali diversi da quelli a cui è abituato l'utente, la diversificazione è un qualcosa di drastico, di rottura rispetto alle soluzioni proposte da altri; un'innovazione del genere richiede all'utente degli elevati costi di apprendimento.
Con "differenziare" si intende, invece, un'operazione meno drastica, si parte dalle soluzioni offerte dagli altri sistemi presenti, dalle applicazioni che l'u-

tente è abituato ad usare; a questo punto si individuano quali potrebbero essere i tratti da differenziare, cioè le funzionalità da implementare in modo diverso così da poter rispondere agli obiettivi che si hanno in mente per la progettazione della nuova applicazione.
Quest'ultima soluzione è stata ritenuta più appropriata e coerente per gli scopi dell'applicazione che si andava a sviluppare: un'applicazione di supporto per i ricercatori che non richiedesse un alto costo di apprendimento, ma che, al contrario, risultasse intuitiva all'utilizzo e semplificasse l'esecuzione di determinate operazioni.

Feedback informativi e messaggi utili

L'utente deve essere costantemente e correttamente informato sull'effetto delle sue azioni sul sistema. Quando l'utente compie un'azione di ricerca deve essere informato sullo stato e sull'esito di quest'ultima: se si stanno caricando i risultati della ricerca, è utile informare l'utente sul progresso di quest'utlima. Se ad esempio la ricerca non dovesse portare ad alcun risultato non è sufficiente mostrare un elenco dei risultati vuoto, bisogna notificare che la ricerca non ha dato risultati; questo è necessario anche per permettere di differenziare altri casi in cui ad esempio non ci siano risultati perchè non è possibile recuperare correttamente le informazioni (ad esempio se un server non è contattabile).
Se l'utente attiva un'opzione che condiziona in qualche modo lo stato delle informazioni visualizzate (ad esempio l'attivazione di qualche tipologia di filtro), egli deve essere correttamente notificato sulle conseguenze di quella azione.
Nel caso in cui si verifichino errori, questi devo essere informativi e non devono disorientare l'utente con messaggi di errore incomprensibili: il linguaggio da usare è quello dell'utente, non quello del sistema.

Dialoghi semplici e design minimalista

Le informazioni da mostrare all'utente devono essere solo quelle fondamentali e rilevanti per il task che si sta eseguendo e bisogna rendere facilmente accessibili e visibili le informazioni necessarie per procedere nell'interazione: se una funzionalità può essere utile all'utente per procedere nell'esecuzione di un task, questa deve essere ben visibile.
Il design delle componenti grafiche deve risultare il più semplice possibile e deve mettere in evidenza il contenuto informativo; ogni componente o abbellimento estetico aggiuntivo va attentamente valutato in quanto non deve risultare pesante a tal punto da mettere in secondo piano l'informazione rilevante per l'utente.
I dialoghi devono essere brevi, non troppo articolati o con troppi livelli di categorie e sottocategorie che portino un'interazione semplice a diventare inutilmente estesa. Tutto questo in un'ottica di riduzione del costo cognitivo.

Seguire il task dell'utente

La sequenza delle operazioni da seguire per giungere ad un obiettivo deve essere il più possibile compatibile con l'evoluzione del task nella mente dell'utente, egli ad ogni interazione con il sistema deve poter individuare subito cosa gli serve per proseguire. La struttura delle informazioni fornite non deve essere legata alla forma con cui le stesse informazioni sono gestite internamente dal sistema, ma deve adattarsi alle esigenze dell'utente: le informazioni devono essere rielaborate, aggregate o divise nel tentativo di presentarle all'utente nella forma che lui si aspetterebbe di trovare.

Informazioni in forma grafica

Quando è necessario che l'utente analizzi una grande quantità di informazioni è preferibile fornirle sotto forma di visualizzazioni grafiche: sintetizzando dati e dettagli con forme, colori e dimensioni è possibile fornire all'utente una rappresentazione di impatto di un certo dominio di dati che gli consente

di interpretare le informazioni fondamentali per mezzo del riconoscimento immediato e naturale di certi pattern e senza la necessità di dover decodificare e comprendere dati la cui forma numerica precisa non è fondamentale in una prima analisi; risulta comunque conveniente fornire all'utente, nel momento in cui lo ritenga necessario, la possibilità di conoscere i dati numerici esatti celati dalla rappresentazione.

Le osservazioni e i principi esposti riassuntivamente in questa sezione hanno rappresentato le principali linee guida nel corso della progettazione; ulteriori dettagli su come essi siano stati effettivamente applicati verranno forniti contestualmente alla descrizione delle singole funzionalità implementate.

3.4 Interfaccia utente e funzionalità

3.4.1 Il layout principale

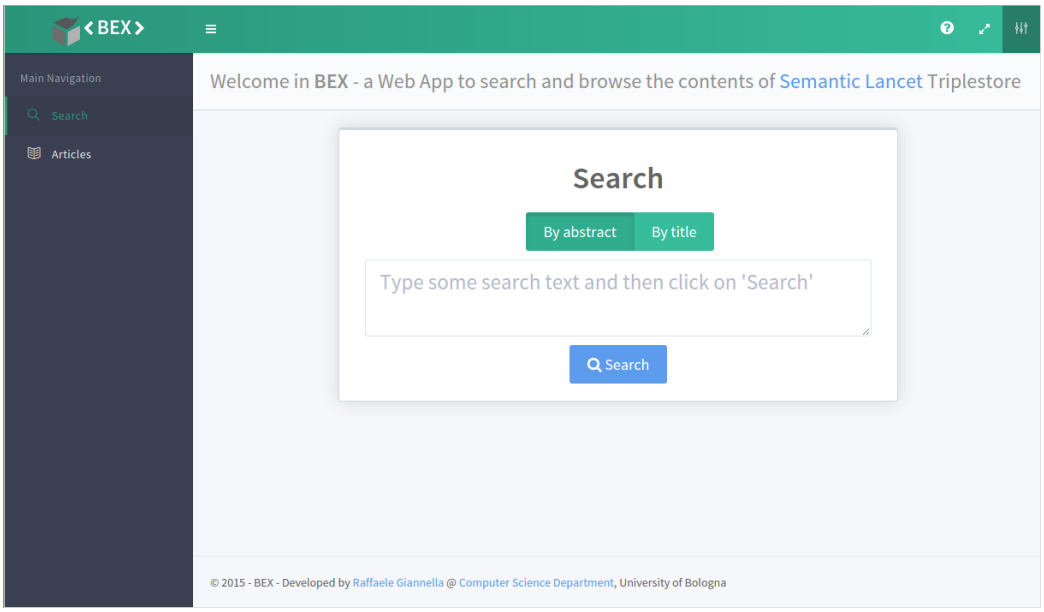

Figura 3.4: Homepage dell'applicazione con navbar e sidebar

Il layout di BEX (Figura 3.4) è molto semplice e minimale nell'aspetto, è composto da 3 componenti principali.

Top navbar

La parte più alta nel layout è occupata da una Top-navbar verde nella quale sono presenti alcuni collegamenti utili da avere a portata di mano durante l'utilizzo dell'applicazione. Procedendo da sinistra verso destra:

- **Logo dell'applicazione**: se cliccato reindirizza alla homepage
- **Bottone menu**: permette di alternare la visualizzazione del sidebar menu in forma estesa (cioè con i labels delle voci di menu) e forma compatta (mostrando solo le icone senza i labels)

- **Bottone guida**: se cliccato reindirizza ad una pagina con una breve guida all'uso dell'applicazione
- **Bottone schermo intero**: permette di visualizzare l'applicazione a tutto schermo dando così più spazio ai contenuti
- **Bottone filtri**: da accesso alla offsidebar con le impostazioni per filtri e ordinamenti sulla bibliografia e permette di capire se sono state attivate delle impostazioni che condizionano la visualizzazione di alcuni elementi informativi nell'elenco dei risultati di ricerca: l'icona stessa ne segnala la presenza

Si è voluto utilizzare questa soluzione della Top-navbar anche nell'ottica di sviluppi futuri dell'app: nel caso vengano implementate nuove funzionalità che necessitano di essere sempre visibili e accessibili all'utente, la Top-navbar rappresenta un ottimo posto nel quale posizionare i bottoni per accedere a queste funzionalità in quanto si adatta bene all'utilizzo come "barra degli strumenti".

Side navbar

Sulla parte sinistra è presente una navbar laterale che ha lo scopo di rendere facile la navigazione verso tutte le principali sezioni (*views*) dell'applicazione. Ogni voce del menu è un collegamento alla view relativa consentendo così di spostarsi da una sezione all'altra; rappresenta inoltre una importante fonte di orientamento per l'utente in quanto permette di capire subito in quale contesto ci si trova. Per le funzionalità attualmente implementate la navbar laterale potrebbe essere ritenuta superflua in quanto sono presenti solo due principali views, cioè "Search" e "Articles", ma c'è una ben precisa motivazione di progettazione per cui è stata inclusa nel layout: facilitare l'implementazione di nuove funzionalità. L'applicazione è destinata ad essere estesa in futuro e molto probabilmente saranno necessarie nuove views da rendere facilmente accessibili all'utente, come ad esempio views per elementi di *data visualizations* o views per l'analisi dettagliata di un certo dominio

di dati non ancora gestito dall'applicazione. Con questa scelta della navbar laterale (e grazie ad alcune scelte di progettazione descritte nel capitolo seguente) si rende semplice e veloce l'aggiunta di queste nuove funzionalità senza la necessità di dover riprogettare l'intera interfaccia per adattarla ai nuovi sviluppi.
Questa barra laterale potrebbe andare a ridurre lo spazio disponibile per i contenuti, soprattutto se la dimensione dello schermo è ridotta, per questo motivo è stata prevista, attraverso il bottone dedicato nella Top-navbar, la possibilità di "collassare" il menu in una forma più compatta; inoltre, su schermi di ridotte dimensioni, il menù stesso scompare dall'interfaccia per non occupare nemmeno un *pixel*, per aprirlo basterà cliccare sul bottone della Top-navbar.

Area principale dei contenuti

In quest'area sono visualizzate tutte le componenti di ogni view disponibile; tutti i contenuti in questa area si adattano automaticamente alla risoluzione dello schermo e non è mai richiesto all'utente di ridimensionare manualmente le componenti o scorrere orizzontalmente per mostrare contenuti non visualizzati (responsive design, discusso nel capitolo seguente).

3.4.2 La ricerca

Nella figura 3.5 è visualizzato il dettaglio del pannello di ricerca. Si sceglie una modalità tra quelle disponibili, si inserisce del testo e cliccando sul bottone "Search" parte la richiesta al server che effettuerà una ricerca basata su quanto inserito. Una volta effettuata tale richiesta ci sarà un cambio automatico della view e si passerà in "Articles" per poter visualizzare gli eventuali risultati di ricerca.
Le modalità di ricerca attualmente disponibili sono le seguenti:

- **By abstract**: si inserisce del testo libero, come frasi o parole chiave a scelta dell'utente; la ricerca verrà effettuata (contattando il servizio

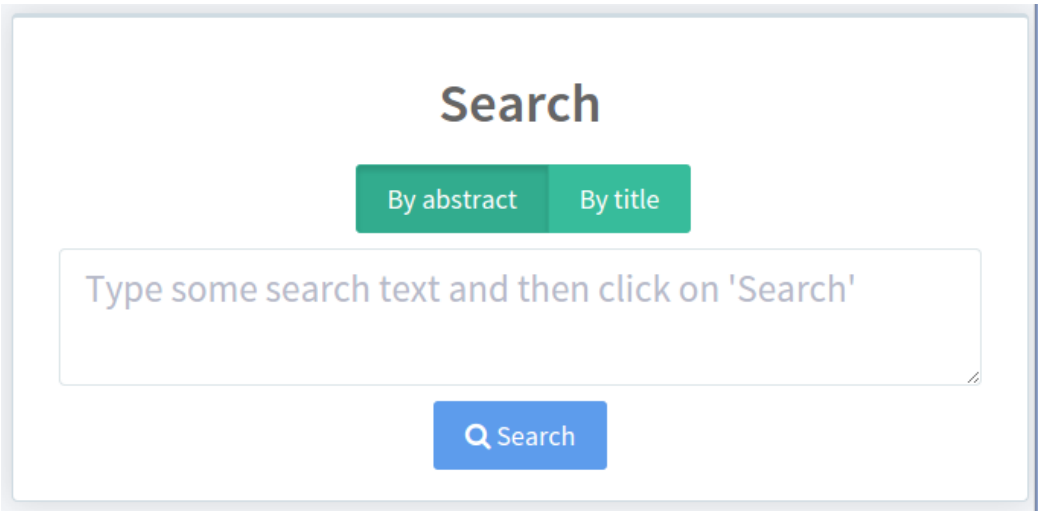

Figura 3.5: Pannello di ricerca

Abstract Finder descritto nel capitolo seguente) sui "concetti" descritti negli abstract degli articoli scientifici nel dataset e i risultati di ricerca saranno composti da un insieme di articoli ritenuti rilevanti ed in qualche modo legati semanticamente a quanto inserito nell'area di testo.

- **By title**: si inserisce un titolo, o parte del titolo di un articolo scientifico; la ricerca verrà effettuata scorrendo i titoli di tutti gli articoli e il risultata sarà composto da un singolo articolo o da una breve lista di articoli nel caso ci siano più papers che contengono nel titolo le parole inserite nella area di testo.

Un placeholder nell'area di testo fornisce le indicazioni sul tipo di testo da inserire in base alla tipologia di ricerca selezionata.

3.4.3 I risultati

Dopo aver eseguito una ricerca ed essere passati alla view "Articles" verrà visualizzata una piccola finestra pop-up che segnala all'utente che sono in corso le attività di ricerca dei contenuti.

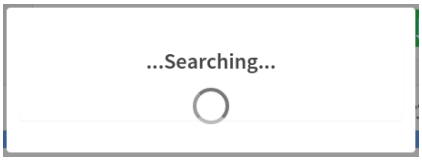

Figura 3.6: Pannello di caricamento risultati

Una volta completata questa operazione
è necessario richiedere al server le informazioni relative ad ognuno degli articoli dei risultati; durante questa fase l'utente sarà informato sul progresso nel caricamento dei risultati attraverso un pannello con una *progress bar* (Figura 3.6) che mostra lo stato delle richieste; questo pannello si chiuderà una volta caricati tutti i risultati.
Nella parte alta della pagina, subito sotto la top-navbar, è presente un pannello di intestazione nel quale viene specificato il contesto dei risultati di ricerca, cioè a quale tipo di ricerca sono legati i risultati mostrati; esso sarà inoltre usato per la *navigazione con breadcrumb* (vedere sezione 3.4.8).
Sotto questa intestazione è presente un pannello centrale che permette di ordinare i risultati di ricerca in base a:

- Anno di pubblicazione
- Titolo
- Citazioni ricevute globalmente[2]

L'ordinamento può essere crescente o decrescente.

[2]fonte: http://www.scopus.com/

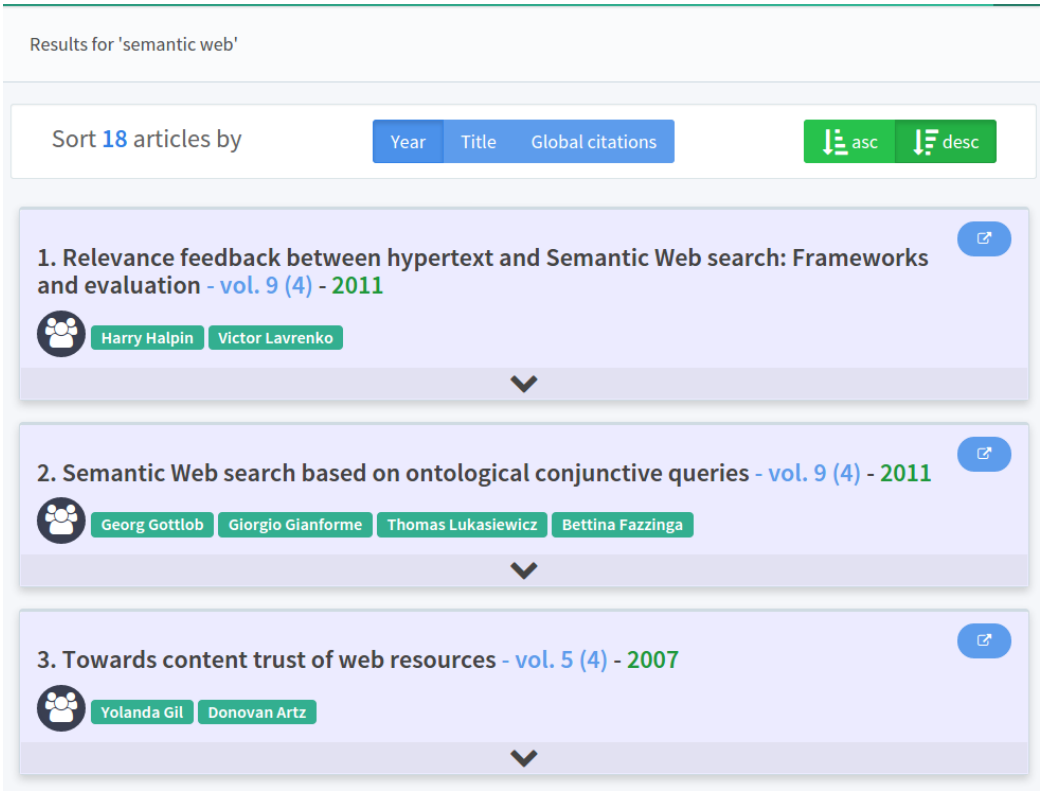

Figura 3.7: Esempio di risultati di ricerca

I risultati di ricerca (esempio in Figura 3.7) sono mostrati con un elenco di articoli da scorrere verticalmente.
In un primo momento, nella fase di progettazione dell'interfaccia, si era pensato di proporre i risultati utilizzando altri modelli di impaginazione e layout, come ad esempio il cosiddetto *masonry layout* nel quale ogni pannello (articolo) va a comporre una sorta di parete di elementi disposti su più colonne e giustapposti tra di loro; in seguito all'implementazione, testando il risultato finale, si è però appurato che tale struttura rendeva poco naturale l'analisi degli articoli; gli utenti, infatti, sono molto più abituati a scorrere risultati disposti in sequenza verticale in quanto, innanzitutto, è un'operazione più naturale da eseguire, ma anche perchè l'utente ricercatore ha più familiarità con tale disposizione, usata nella maggioranza delle DL. Questo è un caso nel quale si è preferito rimanere fedeli agli standard per non richiedere all'utente

un inutile sforzo di apprendimento; si è optato quindi per la disposizione verticale.

Nessuna paginazione per i risultati

Come è possibile notare utilizzando l'applicazione, non è presente alcun meccanismo di paginazione dei risultati di ricerca; è stata fatta questa scelta in quanto, considerando i tasks eseguiti dal ricercatore, si è ritenuta tale funzionalità non necessaria se non addirittura controproducente. Perchè questa scelta? perchè non c'è decrescita esponenziale dell'interesse dell'utente verso un articolo, nel passaggio da un risultato al successivo nella lista; detto in altri termini: la lista dei risultati che viene proposta all'utente è un insieme di articoli i quali hanno tutti un livello di rilevanza (rispetto ai termini di ricerca) sostanzialmente equivalente e non ci sono articoli che sono nettamente meno rilevanti di altri.
Si pensi ad esempio ad una ricerca effettuata utilizzando un motore di ricerca come Google: i risultati nella prima pagina sono molto più pertinenti rispetto a quelli nella seconda, ha quindi senso "tagliare" i risultati per mettere in evidenza quelli sicuramente più rilevanti.
Nell'applicazione sviluppata, perlomeno allo stato attuale, non c'è questa esigenza; si è quindi preferito non forzare l'utente a doversi muovere su più pagine per poter visualizzare tutti i risultati, anche perchè solitamente il numero di articoli proposti dopo una ricerca è abbastanza contenuto da poterli mostrare in una sola pagina.
In futuro, se dovessero cambiare i parametri discussi precedentemente, sarà valutata la possibilità dell'implementazione di tale funzionalità.

3.4.4 Il singolo risultato

Per mostrare le informazioni sul singolo articolo dei risultati di ricerca è stata appositamente progettata una componente grafica che permette di assecondare l'utente nell'esecuzione dei suoi tasks e non richiedere un elevato costo cognitivo. Alcune delle idee per l'implementazione sono state

tratte dalla funzionalità di *drill-down* ampiamente usata in software di Business Intelligence e dal cosiddetto *Shneiderman's Information Seeking Mantra*: "Overview first, zoom and filter, then details on demand"[SHN96]

Drill-down

Per *drill-down* si intende il passaggio da un livello di informazioni generiche o aggregate ad un livello in cui vengono mostrati maggiori dettagli informativi. Questo passaggio avviene su richiesta dell'utente tramite un'interazione (click) con l'elemento di cui si ha intenzione di approfondire l'analisi: si pensi ad esempio ai *menu dropdown* o alla navigazione attraverso le varie directory di un file system utilizzando un file manager.
Nell'ambito dei software di Business Intelligence la funzione di drill-down permette, in pratica, di "esplodere" un dato aggregato, per esempio un indice di bilancio mostrato in un report, per analizzarne le singole componenti, ad esempio i singoli dati che passati ad una certa funzione permettono di ottenere il dato aggregato inizialmente visualizzato.
Qual è il vantaggio di un tale approccio? l'utente non deve interpretare una grande quantità di dati numerici, ma osservare dei semplici indicatori aggregati, magari visualizzati in forma grafica, per poi approfondire solo se lo ritiene necessario; ciò, inoltre, ha il vantaggio pratico di consentire di rendere l'interfaccia più pulita e leggibile. Qual è il rischio di un tale approccio? abusando di tale funzionalità si rischia di rendere l'interazione più lunga e complessa del necessario: bisogna evitare che l'utente sia costretto a muoversi ripetutamente in profondità nella "gerarchia" delle informazioni per reperire i dati ritenuti rilevanti

Shneiderman's Information Seeking Mantra

"*overview first, zoom and filter, then details on demand*"[SHN96]
Si tratta di una linea guida di *visual design* che spiega come i dati andrebbero presentati nella maniera più efficiente per l'utente:

- **Overview**: permettere all'utente di avere una panoramica su un insieme di risultati
- **Zoom**: dare all'utente la possibilità concentrarsi su alcuni elementi di interesse
- **Filter**: fornire all'utente funzioni di *filtering* per escludere informazioni ritenute non rilevanti
- **Details on demand**: quando l'utente lo ritiene necessario, egli deve poter approfondire l'analisi di alcuni dati ricevendo maggiori dettagli

Riassumendo: l'obiettivo principale è quello di seguire l'utente nell'esecuzione del suo task mostrando, ad ogni interazione, le informazioni che egli presumibilmente ritiene rilevanti per l'attività che sta eseguendo; non sovraccaricarlo con ulteriori dettagli informativi che egli deve comunque poter ottenere facilmente, ma solo nel momento in cui ritiene che gli siano utili.
Per questo motivo la componente del singolo articolo dei risultati di ricerca è stata progettata per suddividere le informazioni su più livelli, in una gerarchia che si muove dal generico allo specifico; il tutto è reso possibile, però, cercando un giusto equilibrio tra questa suddivisione delle informazioni e il numero di interazioni necessarie all'utente per ottenere i dettagli, deve cioè essere possibile giungere ad un'analisi dettagliata di alcuni dati senza dover effettuare troppi passi nella navigazione e senza dover cambiare troppe volte il contesto. Nelle successive sezioni viene descritto come siano state divise le informazioni relative ad un articolo scientifico.

Informazioni generiche

In Figura 3.8 è rappresentato un singolo risultato di ricerca, è possibile notare come le informazioni che vengono fornite a questo livello siano solo quelle generiche, pochi elementi essenziali che permettono di identificare l'articolo e di farsi una prima idea su ciò di cui tratta.
In particolare, vengono mostrati:

Figura 3.8: Informazioni generiche su un articolo

- **Titolo**
- **Volume e Issue** (non sempre presenti)
- **Anno di pubblicazione**
- **Lista degli autori**
- **Link esterno** all'articolo nella digital library in cui esso è stato effettivamente pubblicato

Vengono mostrate queste informazione nel "primo livello" in quanto si ritiene che siano questi gli elementi che vengono osservati dall'utente in una prima analisi dei risultati di una ricerca; in base ad essi il ricercatore può arrivare ad una prima selezione degli articoli ritenuti interessanti e di cui intende approfondire l'analisi. Cliccando sulla barra nella parte bassa della componente è possibile mostrare altri dettagli sull'articolo.

Dettagli su un articolo

In Figura 3.9 è mostrato un articolo in cui vengono visualizzate le informazioni generiche ed un primo livello di dettagli fondamentali per un'analisi più approfondita.
In particolare vengono fornite le seguenti informazioni:

- **Abstract**: all'interno di un pannello dedicato è mostrato l'abstract dell'articolo; questo è uno degli elementi fondamentali per l'analisi di un articolo in quanto permette di farsi un'idea sulle tematiche trattate nel paper. Inoltre, una volta letto l'abstract, è possibile "richiudere"

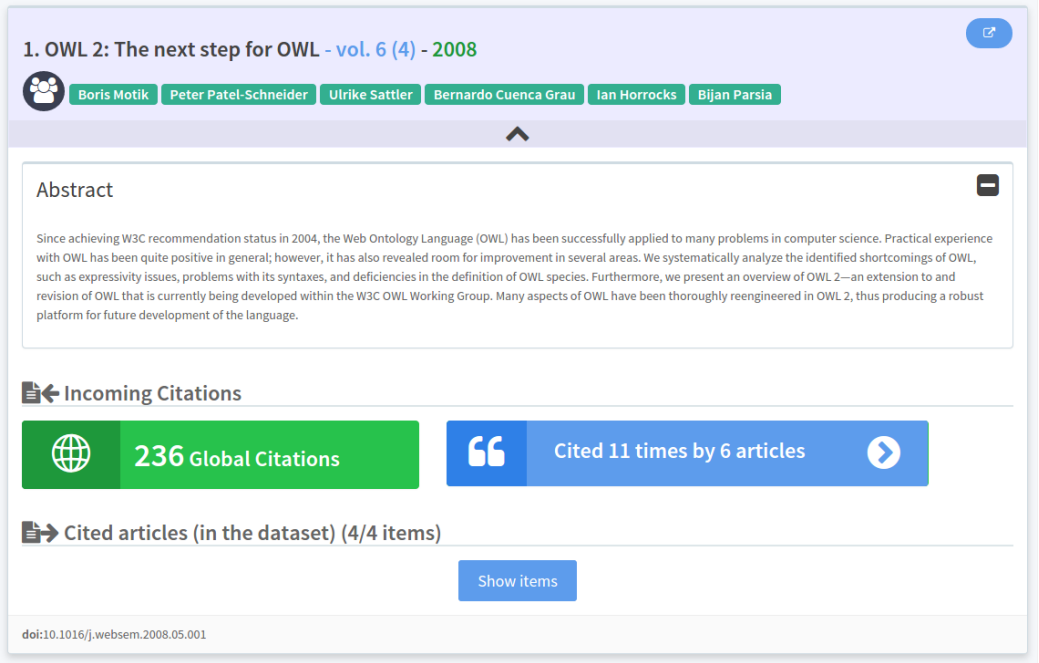

Figura 3.9: Dettagli su un articolo

il pannello in modo da non occupare spazio nell'interfaccia; questo per fare in modo che la *scroll view* nella quale sono mostrati tutti i risultati di ricerca non diventi eccessivamente lunga e di conseguenza scomoda da scorrere per l'utente

- **Sezione Incoming Citations** (citazioni ricevute): in questa sezione sono mostrati due widgets. Il primo (di colore verde) mostra il dato aggregato sul numero di citazioni globali ricevute dall'articolo, il secondo (di colore blu) visualizza 2 indicatori aggregati sul numero di citazioni ricevute da un certo numero di articoli presenti nel dataset. Le citazioni globali[3] sono mostrate con un indicatore aggregato che non può essere ulteriormente approfondito in quanto nel dataset non sono attualmente disponibili informazioni in merito. Riguardo al secondo widget, le informazioni fornite rappresentano degli indicatori aggregati il cui studio può essere ulteriormente approfondito: tali indicatori sono

[3]fonte: http://www.scopus.com/

infatti ricavati da informazioni più dettagliate disponibili nel dataset. Per i dettagli su come visualizzare tali informazioni vedere la sezione 3.4.5

- **Sezione Cited Articles** (bibliografia): in questa sezione è possibile analizzare i singoli elementi della bibliografia dell'articolo, cioè gli articoli che esso cita; in un primo momento tali elementi, se presenti, non sono visualizzati singolarmente in quanto potrebbe essercene una discreta quantità (quindi occupare molto spazio nell'interfaccia) e l'utente potrebbe non essere interessato alla loro analisi nel dettaglio. Nell'intestazione della sezione viene mostrato un rapporto tra gli articoli visualizzabili attualmente e il numero totale di articoli citati; perchè è presente questo rapporto? perchè potrebbero essere stati impostati dei filtri sulla bibliografia (discussi più avanti) che vanno ad escludere alcuni elementi dall'elenco degli articoli della bibliografia; è quindi necessario segnalare all'utente che gli elementi visualizzati potrebbero non essere tutti.
  Subito sotto l'intestazione, se ci sono articoli visualizzabili, è presente un bottone "Show items", cliccandoci verrà mostrato l'elenco degli articoli della bibliografia, rappresentati utilizzando un widget appositamente progettato (vedere sezione 3.4.6).

- **DOI**: nel footer del pannello dell'articolo è mostrato il suo Digital Object Identifier

Cliccando nuovamente sulla barra nel *header* del pannello (informazioni generiche) è possibile nascondere nuovamente i dettagli relativi all'articolo.

3.4.5 Citazioni ricevute

Analizzando un articolo è possibile approfondire lo studio delle citazioni che esso ha ricevuto da altri articoli presenti nel dataset. Nella sezione "*Incoming Citations*" di cui si è parlato in precedenza è presente un widget come quello mostrato in Figura 3.10; in questo widget, se sono presenti citazioni, è possibile cliccare su un bottone che permetterà di aprire una finestra modale[4] che mostra utili informazioni relative alle citazioni in entrata.

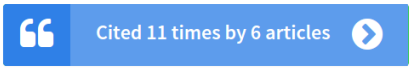

Figura 3.10: Widget per le citazioni in entrata

La finestra modale è divisa in due sezioni.

Overview on incoming citations

In questa prima sezione viene fornito uno sguardo d'insieme sulle tipologie di citazioni ricevute dall'articolo.
In particolare vengono visualizzati due grafici interattivi.

Il primo grafico, mostrato in Figura 3.11, è un *Donut chart* che permette all'utente di avere un rapido quadro sui motivi per cui l'articolo è stato citato, cioè i motivi citazionali.
Per ogni componente del grafico viene visualizzata un'etichetta che ne esplicita il *citation function*[5]; è inoltre possibile sapere il numero preciso di cita-

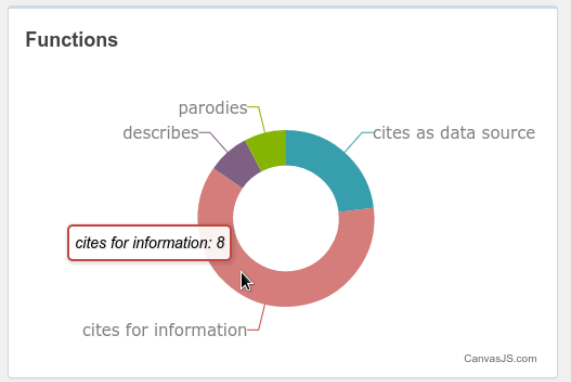

Figura 3.11: Donut chart

[4]una finestra all'interno dell'applicazione
[5]motivo citazionale

zioni, posizionandosi con il mouse sul singolo componente. Con questo grafico l'utente può capire che reazione in generale ha avuto la comunità scientifica verso l'articolo preso in esame; ad esempio permette di rispondere rapidamente a domande del tipo "é stato esteso?", " è stato criticato?", " è stato usato come fonte di informazioni?", "ciò di cui tratta è stato confermato da altri?".

Il secondo grafico, mostrato in figura 3.12, è uno *Stacked column chart* che sostanzialmente visualizza le stesse informazioni del *donut chart* precedente, ma lo fa da un'altra prospettiva: aggiunge la dimensione temporale alla rappresentazione, permettendo all'utente di osservare come le citazioni siano cambiate nel tempo, individuando così delle tendenze nell'evoluzione delle reazioni verso l'articolo; ad esempio permette di fare osservazioni del tipo "le citazioni nel corso degli anni sono aumentate in modo esponenziale, l'articolo sta avendo un grande impatto", "le citazioni sono aumentate, ma sono aumentate anche le critiche, forse sono state evidenziate delle crepe nell'analisi condotta nell'articolo", "le citazioni sono tendenzialmente in crescita e molte di esse sono per estensioni o aggiornamenti, si sta formando una comunità che si occupa della tematica affrontata nell'articolo".

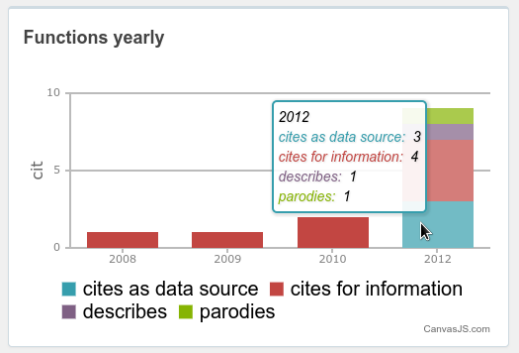

Figura 3.12: Stacked column chart

L'utente può quindi valutare la bontà dell'articolo sotto certi punti di vista, analizzando il tipo di citazioni ricevute.

Per ricavare l'informazione sull'anno in cui è stata ricevuta una certa citazione è stato preso in considerazione l'anno di pubblicazione dell'articolo che effettua quella citazione.

Anche in questo caso è possibile visualizzare il dettaglio del numero di

citazioni posizionandosi con il mouse su una delle barre del grafico.

Cited by (in the dataset)

La seconda sezione della finestra modale è composta dalla lista degli articoli che citano il paper preso in esame.
L'utente ha così accesso rapido alle informazioni essenziali relative agli articoli citanti e, nel caso lo ritenga necessario, può approfondire l'analisi di un articolo (funzione descritta nella sezione 3.4.8)
Anche in questo caso le informazioni relative al singolo articolo citante sono mostrate utilizzando una componente grafica appositamente progettata.

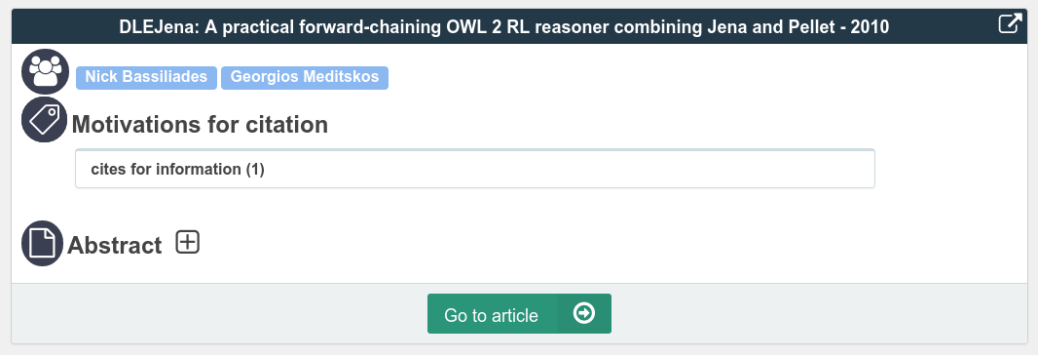

Figura 3.13: Esempio di articolo citante

In Figura 3.13 è rappresentato un esempio di articolo citante; vengono mostrate alcune informazioni essenziali, cioè il titolo, l'anno di pubblicazione, gli autori e l'abstract, quest'ultimo di default è chiuso per non occupare troppo spazio nell'interfaccia. Anche in questa componente è presente un collegamento che fornisce un link esterno all'articolo nella DL in cui è stato pubblicato.
La sezione che contraddistingue questa componente è "*Motivations for citation*": qui viene mostrato l'elenco dei motivi per cui l'articolo cita il paper analizzato[6].
Per ogni motivo viene indicato tra parentesi il numero di volte che l'articolo

[6]quello dei risultati di ricerca

cita per quel motivo e cliccando sull'intestazione (vedere Figura 3.14) vengono mostrati i contesti citazionali, cioè le frasi, all'interno dell'articolo citante, nelle quali viene menzionato l'articolo analizzato; questo è un valore aggiunto rispetto alle semplici informazioni citazionali in quanto permette di leggere in che modo e con quali parole l'articolo citante si riferisce all'articolo citato. Nel caso in cui l'articolo citante abbia degli autori (almeno uno) in comune con l'articolo citato questi verranno evidenziati in colore verde; inoltre l'articolo sarà facilmente riconoscibile come "autocitazione" grazie alla presenza di una icona con una "S" e con lo sfondo dell'intestazione di colore viola, come illustrato in Figura 3.14. Risulta utile sottolineare questa tipologia di articoli in quanto permette di effettuare rapidamente delle osservazioni su come gli autori dell'articolo originale[7] abbiano portato avanti le tematiche o i progetti esposti; è possibile ad esempio capire come si sia evoluto un certo progetto nel tempo, se gli autori ci abbiano lavorato in modo costante o se magari il progetto abbia subito una battuta d'arresto.
In ogni articolo citante è presente un bottone "Go to article" che permette di passare all'analisi dettagliata dell'articolo; si rimanda la discussione alla sezione 3.4.8

Figura 3.14: Esempio di articolo citante con autori condivisi

[7] l'articolo dei risultati di ricerca

3.4.6 Elemento della bibliografia

Figura 3.15: Esempio di elemento della bibliografia

In ogni articolo dei risultati di ricerca è presente una sezione dedicata agli articoli della bibliografia, cioè gli articoli citati.
Questi elementi sono molto importanti in quanto, come detto in precedenza, consentono di comprendere meglio su quali argomenti gli autori dell'articolo abbiano lavorato, a quali lavori si siano ispirati per la loro trattazione; essi, inoltre, rappresentano per l'utente uno dei principali "ponti" di collegamento con altri articoli che trattano tematiche affini e il cui studio potrebbe essere rilevante per l'esecuzione di determinati tasks.
Per gli elementi della bibliografia è stata ideata una componente grafica con l'intenzione di rispondere a diverse esigenze.
L'elemento deve essere facilmente leggibile, compatto e deve essere abbastanza informativo da fare in modo che l'utente possa fare delle prime considerazioni significative.
In Figura 3.15 è illustrato un esempio di elemento della bibliografia.
Il pannello colorato sulla sinistra mette in evidenza il numero di citazioni ricevute dall'articolo citante, nella parte centrale troviamo i seguenti elementi:

- **Titolo**
- **Anno di pubblicazione**
- **Link esterno**
- **Elenco degli autori**

- **Numero di citazioni globali**[8]

- **Bottone Abstract**: permette di visualizzare l'abstract in una finestra pop-up evitando di occupare ulteriore spazio nell'interfaccia

- **Bottone Citation functions**: permette di visualizzare l'elenco dei motivi citazionali con annessi contesti citazionali; anche queste informazioni sono mostrate in una finestra pop-up

- **Donut chart dei motivi citazionali**: mostra graficamente i motivi per cui l'articolo dei risultati di ricerca cita l'elemento bibliografico preso in esame (il totale è uguale al numero di citazioni del pannello sulla sinistra); per questo grafico si possono fare considerazioni analoghe a quelle fatte in precedenza per il primo grafico nella sezione 3.4.5 con la differenza che in questo caso i motivi sono relativi al singolo elemento bibliografico

Anche in questo caso, come nel caso degli articoli citanti, se ci sono autori condivisi con l'articolo dei risultati di ricerca, essi saranno evidenziati con colore verde e l'intero elemento bibliografico sarà riconoscibile rapidamente come "autocitazione" grazie allo sfondo viola del pannello sulla sinistra e per l'icona con la "S" (vedere Figura 3.16). Infine, sulla parte destra dell'elemento bibliografico è presente un bottone con una freccia, esso permette di approfondire l'analisi dell'elemento bibliografico passando a visualizzare i dettagli dell'articolo, questa funzionalità verrà illustrata nella sezione 3.4.8. Da notare come il layout dell'elemento sia stato strutturato in modo da occupare il più possibile la dimensione orizzontale: questa soluzione ha il vantaggio di rendere più compatta la visualizzazione dell'elenco degli articoli in una bibliografia e permette inoltre di mettere in evidenza alcune parti dell'elemento come il numero di citazioni ricevute (dall'articolo citante), il donut chart dei motivi citazionali e il bottone per l'approfondimento dell'articolo.

[8]fonte: http://www.scopus.com/

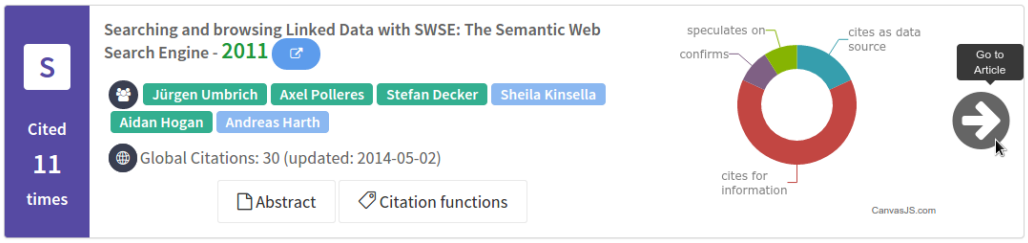

Figura 3.16: Esempio di autocitazione nella bibliografia

3.4.7 Filtri e ordinamenti sulla bibliografia

Figura 3.17: Offsidebar dei filtri sulla bibliografia

Dall'analisi dei tasks e delle esigenze dei ricercatori emerge come la bibliografia sia una delle sezioni sui cui il ricercatore passa più tempo.
Si è pensato dunque a come rendere più facile ed immediata l'analisi degli elementi bibliografici; prendendo in considerazione alcuni dei principi enunciati nelle precedenti sezioni, come ad esempio lo "*Shneiderman's Information Seeking Mantra*"(3.4.4), e attraverso il confronto diretto con alcuni ricercatori,

si è giunti alla conclusione che sarebbe stato molto utile fornire delle funzioni che rendessero possibili operazioni di *filtering* sugli elementi della bibliografia in modo da mostrare solo gli elementi ritenuti rilevanti per il task che l'utente sta eseguendo: se l'utente ha la necessità, ad esempio, di esaminare solo le autocitazioni di più articoli, sarebbe comodo avere una funzionalità che mostri solo tali elementi, nascondendo tutti gli altri. Sono allora stati implementati i filtri per la bibliografia, accessibili all'utente dal bottone presente sulla destra della top-navbar; cliccando su tale bottone si aprirà una *offsidebar* sul lato destro della finestra (Figura 3.17), cioè un pannello "a scomparsa" che andrà a coprire temporaneamente parte dei contenuti visualizzati. Questa offsidebar, sostanzialmente, si trova ad un livello (*layer*) più alto rispetto ai risultati di ricerca e quando aperta si sovrappone a tali risultati; è stata scelta questa soluzione per evitare di dover ridimensionare i contenuti sottostanti per doversi adattare ad uno spazio ridotto; con questa soluzione del pannello "a scomparsa" si permette di usare i filtri anche su schermi di dimensioni ridotte.
La offsidebar è suddivisa in due *tabs*: una tab per i filtri ed una per definire l'ordinamento da usare nelle bibliografie degli articoli. Per passare da una tab all'altra è sufficiente cliccare sulle due icone presenti nella parte alta del pannello.
Per attivare un filtro è necessario cliccare sul bottone *switch* posizionato alla destra del nome del filtro, verranno dunque mostrati eventuali elementi per impostare il filtro; per disattivare un filtro basta cliccare nuovamente su tale *switch*.
Un punto fondamentale da evidenziare: una volta attivato un filtro esso non verrà applicato alla sola bibliografia dell'articolo analizzato, ma alla bibliografia di tutti gli articoli visualizzati; in questo modo si cerca di garantire un'esperienza d'uso più coerente con le intenzioni del ricercatore che sta eseguendo un determinato task. L'utente non dovrà perdere tempo a reimpostare i filtri nel passaggio da un articolo all'altro, la bibliografia di tutti gli articoli sarà già filtrata secondo i parametri da lui stabiliti; per evitare

confusione l'utente verrà comunque notificato, in modo intuitivo e non invasivo, che la visualizzazione degli elementi della bibliografia potrebbe essere condizionata dai filtri attivati.
Allo stato attuale l'applicazione mette a disposizione le tipologie di filtro qui di seguito descritte; nella loro descrizione, parlando di "articoli" ci si riferisce agli articoli della bibliografia.

Anno di pubblicazione (Year)

Attivando questo filtro e impostando un anno, nella bibliografia verranno visualizzati solo gli articoli con anno di pubblicazione maggiore o uguale all'anno selezionato. Questo filtro risulta utile quando si vogliono analizzare solo articoli di recente pubblicazione.

Solo autocitazioni (Only self-citations)

Questo semplice filtro, una volta attivato, permette di mostrare solo gli articoli che hanno almeno un autore condiviso con l'articolo citante, cioè le autocitazioni. Visualizzando le sole autocitazioni, l'utente può concentrarsi sullo studio di come gli autori abbiano lavorato su una certa tematica o su un progetto.

Autori (Authors)

Per mezzo di questo filtro l'utente può fare in modo che nella bibliografia siano elencati solo gli articoli scritti da un certo autore o da un insieme di autori; questo filtro è utile quando l'utente vuole concentrare il suo studio sull'attività di un certo autore (o un ristretto insieme di autori) e analizzare quanto trattato nei suoi (o loro) lavori.
Una volta attivato il filtro è sufficiente cominciare a scrivere il nome nella casella di testo per avere una lista dei possibili autori; è stata implementata questa funzionalità per scongiurare possibili errori di battitura che andrebbero ad invalidare l'elenco di articoli mostrati nella bibliografia.

Dopo aver selezionato un autore e cliccato su "Add", nella lista subito sotto la casella di testo, verrà aggiunto un *label* con il nome dell'autore; sarà così possibile visualizzare in ogni momento la lista degli autori filtrati e, cliccando sull'icona "x" posta alla sinistra del nome si eliminerà l'autore selezionato dal filtro.

Motivi citazionali (Functions)

Questo filtro permette di visualizzare gli articoli in bibliografia sulla base dei motivi per cui essi sono stati citati dall'articolo dei risultati di ricerca. Attivato il filtro verrà mostrato un pannello con l'elenco di tutti i motivi citazionali disponibili, essi saranno tutti selezionati di default; la selezione dei motivi è di tipo "inclusivo" e non "esclusivo": ciò vuol dire che, se un motivo "X" è selezionato, in bibliografia verranno sicuramente mostrati tutti gli elementi citati per il motivo "X", ma se un altro motivo "Y" è deselezionato questo non vuol dire che dalla bibliografia verranno esclusi gli articoli citati per il motivo "Y"; è stata adottata questa soluzione per evitare i problemi che potrebbero sorgere da una combinazione conflittuale di motivi citazionali.
Per questo filtro si è preferito mostrare subito tutte le possibili alternative, diversamente dal filtro degli autori per il quale è previsto che l'utente inserisca manualmente il nome, il motivo di questa scelta è semplice: un utente che utilizza l'applicazione per la prima volta, probabilmente non ha idea di cosa siano i motivi citazionali e non saprebbe nemmeno cosa scrivere in una eventuale casella di testo; fornendogli una lista completa di tutti i motivi egli può meglio orientarsi tra le alternative e scoprire quali siano i possibili motivi citazionali. Sono inoltre presenti due bottoni che permettono di selezionare o deselezionare rapidamente tutti i motivi citazionali.

Nel momento in cui l'utente attiva almeno uno dei filtri descritti è possibile notare un cambiamento nell'icona sulla destra della top-navbar (vedere Figura 3.17): questo serve a segnalare all'utente, una volta chiusa la offsidebar, che ci sono dei filtri attivati. Si tratta di un'accortezza che è stata ritenuta necessaria dopo aver fatto alcune considerazioni sull'interfaccia: le funzionalità di filtering sono state progettate per essere sempre facilmente accessibili all'utente, ma senza che esse siano perennemente presenti nell'interfaccia mostrata; questo perchè l'utente nella maggior parte dei casi non si troverà ad utilizzare i filtri più volte in poco tempo, ma li utilizzerà saltuariamente per impostare alcuni parametri per la visualizzazione degli elementi della bibliografia. Una volta chiusa la offsidebar, dopo alcune interazioni con l'applicazione, l'utente potrebbe "dimenticarsi" di aver impostato dei filtri di cui magari non ha più bisogno e per questo motivo si ritroverebbe ad analizzare bibliografie in cui alcuni elementi potenzialmente rilevanti non vengono mostrati.
Inoltre, qualora i filtri vadano a modificare l'elenco degli articoli di una bibliografia (evento plausibile), nell'intestazione della sezione della bibliografia verranno mostrate alcune informazioni rilevanti, come illustrato nel dettaglio in Figura 3.18

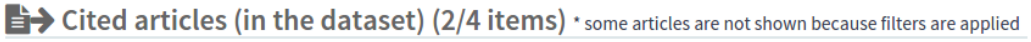

Figura 3.18: Dettaglio dell'intestazione della bibliografia

Ordinamento

Nella offsidebar è presente anche una seconda tab con la quale è possibile andare ad impostare l'ordinamento degli elementi della bibliografia, le opzioni disponibili, come visualizzato in Figura 3.19, sono 4:

- **Anno di pubblicazione** (opzione default)

- **Titolo**
- **Citazioni globali**
- **Citazioni ricevute dall'articolo citante**

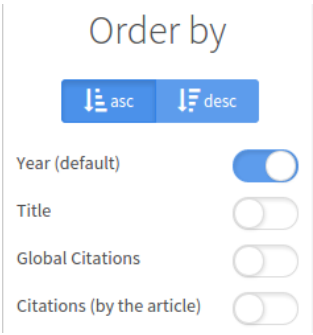

Figura 3.19: Ordinamento degli elementi della bibliografia

Per ogni opzione è possibile inoltre scegliere se l'ordinamento debba essere in senso crescente o decrescente.

3.4.8 Navigazione

L'applicazione fornisce anche alcune fondamentali funzionalità di navigazione per consentire al ricercatore di approfondire lo studio di determinati elementi ritenuti interessanti e rilevanti per il task che sta eseguendo.

Navigazione tra gli articoli

Scelto un articolo ritenuto rilevante, l'utente molto spesso ha interesse ad approfondire lo studio degli articoli che fanno parte della sua rete citazionale (come discusso nei capitoli iniziali di questa trattazione): gli articoli citati e gli articoli che lo citano. Per fornire all'utente questa modalità di navigazione, l'applicazione prevede apposite funzionalità.
In ogni elemento della bibliografia e in ogni elemento citante (nella finestra modale delle citazioni) è presente un bottone che permette di approfondire

lo studio dell'elemento selezionato, cliccando questo bottone verrà effettuata una nuova richiesta al server per ottenere ulteriori informazioni sull'articolo e nell'applicazione si passerà direttamente alla visualizzazione di questi dettagli, come se fosse stata effettuata una nuova ricerca.
Vengono quindi visualizzate ulteriori informazioni sull'articolo, come le informazioni citazionali e la bibliografia; l'utente può, ancora una volta, muoversi verso un altro articolo della sua bibliografia o degli articoli citanti.
Immaginando ogni articolo del dataset come un nodo e le citazioni (in entrata e in uscita) come archi che collegano i nodi, tramite la funzionalità di navigazione implementata, in definitiva, il ricercatore è libero di muoversi in profondità all'interno della rete che si viene a creare, spostandosi di articolo in articolo.

Navigazione per autore

Un'altra modalità di navigazione che si è ritenuto utile implementare è costituita dalla navigazione per autore. A partire da un articolo, il ricercatore potrebbe essere interessato ad uno dei suoi autori e vorrebbe studiarne meglio l'attività di ricerca.
Gli autori nell'intestazione di ogni articolo hanno anche una funzione di collegamento: cliccando su un nome verranno infatti richiesti al server tutti gli articoli scritti da quell'autore e verranno mostrati come risultati di ricerca.
In realtà questa funzionalità non è presente solo per gli autori che compaiono nell'intestazione di un articolo, ma ovunque ci sia un elenco di autori, che si tratti di un elemento della bibliografia o di un articolo citante.
Riprendendo la metafora dei nodi e degli archi utilizzata in precedenza si può dire che, con questa funzionalità, si ha a disposizione un ulteriore livello di collegamento tra i nodi: a partire da un qualsiasi articolo (nodo) l'utente può accedere rapidamente a tutti gli altri articoli di tutti gli autori dell'articolo di partenza, ciò permette di esplorare ancora più efficientemente il dataset dei contenuti digitali disponibili.

Breadcrumb

Nelle due sezioni precedenti sono state illustrate le due modalità di navigazione attualmente disponibili nell'applicazione: l'utente, partendo da un insieme di risultati di ricerca, può in seguito continuare nell'esplorazione dei contenuti digitali e arrivare ad analizzare articoli molto "distanti" dai risultati di ricerca iniziali; sorgono quindi delle problematiche: come fare in modo che l'utente sappia sempre dove si trova e non si senta disorientato? come permettergli di tenere traccia del percorso seguito durante la navigazione? come consentirgli di ripercorrere a ritroso il percorso seguito? Per rispondere a queste domande è stata adottata una semplice soluzione per tenere traccia delle attività eseguite dall'utente durante l'esplorazione: il *Breadcrumb*.
Il breadcrumb (letteralmente "molliche di pane") è un sistema di aiuto alla navigazione utilizzato in molti contesti: è costituito da un menu o più spesso da una barra solitamente posta nella parte alta dell'interfaccia; essa permette di visualizzare i passi seguiti durante la navigazione e fornisce i collegamenti per tornare indietro. Un esempio di breadcrumb è raffigurato nella Figura 3.20; in questo esempio si può notare come sia rappresentato il percorso seguito: è stata inizialmente effettuata una ricerca partendo dalle parole chiave "semantic web", dopodichè si è passati all'analisi dell'articolo "From SHIQ and RDF..." (pubblicato nel 2003), in seguito sono stati richiesti tutti gli articoli di "Ian Horrocks" per spostarsi infine sull'articolo "Reducing OWL entailment...". Ognuno dei riferimenti presenti nel breadcrumb è anche un collegamento che permette all'utente di ritornare alla visualizzazione degli articoli mostrati ad un certo passo nella navigazione; grazie a questa soluzione l'utente sa sempre dove si trova, come ci è arrivato e ha la possibilità di tornare indietro in qualsiasi momento, per seguire percorsi alternativi.

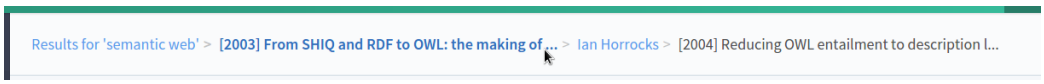

Figura 3.20: Breadcrumb

Guide e tooltips

Per consentire all'utente di prendere familiarità con le funzioni offerte dall'applicazione e con gli elementi visualizzati nell'interfaccia, sono state previste diverse tipologie di guida. Innanzitutto nella top navbar, sulla destra, è sempre presente un collegamento ad una guida rapida all'uso, cliccando su tale collegamento verrà aperta una nuova tab nel browser e verrà visualizzato un breve documento che fornisce una rapida panoramica delle funzionalità offerte da BEX. Oltre alla guida all'uso, sono stati previsti numerosi tooltips[9] informativi; nel caso in cui non si intuisca immediatamente la funzione di un particolare bottone o un altro elemento grafico, è sufficiente posizionarsi con il mouse su tale elemento per visualizzare un breve testo esplicativo.

[9] piccoli fumetti

Capitolo 4

BEX: dettagli implementativi

In questo capitolo verranno analizzati i principali aspetti implementativi che hanno caratterizzato la fase di progettazione e sviluppo. Saranno illustrati e descritti librerie e frameworks utilizzati, per poi passare ad esaminare le principali scelte implementative effettuate e la struttura generale dell'applicazione.

4.1 Tipo di applicazione sviluppata e obiettivi di progettazione

BEX è stata sviluppata come una Single Page Application: una web app che risiede completamente su una singola pagina e nella quale la maggior parte delle interazioni con l'utente sono gestite lato client (browser) senza la necessità di contattare alcun server per caricare elementi dell'interfaccia o convalidare dati.
Questa soluzione permette di rendere interattiva e dinamica l'applicazione garantendo un'esperienza d'uso più fluida. L'utente che accede al sito non carica la sola homepage, nel suo browser viene caricata un'intera applicazione web con tutto il codice Javascript, HTML e CSS necessario per il suo corretto funzionamento; tutte le risorse aggiuntive, come le informazioni sugli articoli, richieste dall'utente durante la navigazione, vengono caricate dina-

micamente dal server utilizzando tecnica AJAX (Asynchronous JavaScript and XML) permettendo di aggiornare solo porzioni della pagina HTML, in modo asincrono e in background; ciò permette di garantire maggiore interattività, minori tempi di attesa e minori quantità di dati scambiati con il server (non è necessario caricare intere pagine). Anche la navigazione all'interno dell'applicazione è gestita completamente client-side: l'utente, nell'utilizzo dell'applicazione, non carica diverse pagine, ma la navigazione è "simulata" con un meccanismo di *routing* che permette di cambiare la view visualizzata all'utente in base alle sue interazioni con il sistema.
Un'applicazione su singola pagina, in definitiva, è considerabile come una sorta di applicazione desktop, composta di codice HTML, javascript e CSS, utilizzabile per mezzo di un qualsiasi browser di ultima generazione; l'intera logica applicativa e di presentazione si sposta dal server al client, il browser utilizzato. Single Page Application non vuol dire che l'applicazione sia talmente semplice nella sua implementazione da poter "girare" su una singola pagina, ma al contrario, una applicazione su singola pagina può essere anche una web app molto complessa e sofisticata al suo interno, questo è il caso di BEX: è necessario quindi dare una struttura a tale applicazione in modo da arrivare ad un risultato di qualità, sotto diversi punti di vista.
Nelle prime fasi della progettazione sono stati fissati alcuni obiettivi da raggiungere per ottenere un buon risultato finale:

1. **Modularità**, **Coerenza** e **Alta Coesione**: l'applicazione deve essere strutturata in modo tale da essere modulare, cioè suddivisa in più componenti ognuna con una singola responsabilità, con un singolo scopo coerente con i dati, le variabili e gli stati che si trova a gestire, in modo da avere un elevato livello di coesione; ogni modulo deve essere legato agli altri secondo uno schema architetturale coerente con le dipendenze che si vengono a creare all'interno del sistema.

2. **Astrazione**: l'applicazione deve essere divisa su più livelli di astrazione, ognuno dei quali "nasconde" i propri dettagli implementativi e

fornisce ai livelli sottostanti un'interfaccia di comunicazione, esponendo un'insieme di funzioni (API).

3. **Estendibilità** e **Manutenibilità**: La misura di quanto un modulo dipenda da altri per il suo funzionamento è definita "accoppiamento"; un basso accoppiamento ed un'alta coesione (punto 1) favoriscono l'estendibilità e la manutenibilità del sistema nel suo complesso. L'obiettivo è quello di rendere possibile effettuare modifiche, estensioni e correzioni in un modulo senza che ciò abbia ripercussioni sugli altri moduli e renda necessario riprogettare altre parti del sistema per adattarsi alle modifiche effettuate.

4. **Riusabilità**: se in più punti del sistema si rende necessario l'utilizzo di una certa funzionalità, allora deve essere disponibile un singolo oggetto, un singolo modulo, un singolo componente o un servizio riusabile che esponga i metodi che permettono di realizzare tali funzionalità; vanno dunque evitate duplicazioni di codice che rendono il sistema più fragile.

5. **Efficienza** e **Robustezza**: alcune delle osservazioni fatte fin qui hanno come diretta conseguenza anche l'incremento della qualità del risultato in termini di efficienza e robustezza; per ottenere ulteriori miglioramenti sotto questi punti di vista è comunque necessario adottare alcune accortezze implementative che derivano dall'attenta osservazione di limiti e mancanze delle tecnologie e dei servizi utilizzati dal sistema per il suo corretto funzionamento.

Per raggiungere questi obiettivi sono stati usati frameworks e librerie per facilitare lo sviluppo e sono state effettuate delle scelte implementative relative alla struttura interna da dare all'applicazione.

4.2 Il principale framework: AngularJS

Per lo sviluppo dell'applicazione, dopo un'attenta analisi delle specifiche funzionali del sistema che si sarebbe andato a realizzare, si è ritenuto utile

l'utilizzo di un framework javascript che permettesse di dare una robusta struttura all'applicazione e assistesse lo sviluppatore nelle fasi di implementazione. Il panorama dei frameworks disponibili per assolvere tali compiti è molto ampio; dopo uno studio delle varie alternative, la scelta è ricaduta su quello ritenuto più completo ed efficiente per la realizzazione dell'applicazione: AngularJS.
Verranno ora descritti i tratti fondamentali di tale framework dopodiché verranno illustrati i principali dettagli implementativi su come esso sia stato effettivamente utilizzato. Per lo studio di tale framework i principali riferimenti utilizzati sono stati [SGR14], [LER13], [GOO15], [THI15], [W3S15]

AngularJS[1] è un MVC framework javascript open-source sviluppato e mantenuto da Google[2] il cui obiettivo è quello di rendere più semplice ed efficiente lo sviluppo client-side di applicazioni web (single page) dinamiche, seguendo il consolidato design pattern *Model-View-Controller*. Questo framework permette di utilizzare HTML come linguaggio di templating e consente di estendere la sintassi di HTML creando elementi e attributi personalizzati. Con angularJS è possibile inoltre stabilire il cosiddetto "data-binding" tra model e view; ma angularJS è molto di più, è quindi utile introdurre i principali concetti in modo riassuntivo.

4.3 MVC

MVC (Model-View-Controller) è un pattern architetturale che consente di separare in modo efficiente la logica di presentazione dei dati dalla business logic consentendo allo sviluppatore di avere un punto da cui partire per stabilire come e a chi assegnare determinate responsabilità all'interno dell'applicazione. Questo pattern architetturale divide l'applicazione in tre parti distinte e modulari:

[1] https://angularjs.org/
[2] https://www.google.com

- **model**: sono i dati gestiti dall'applicazione, generalmente recuperati dal server; tutti i dati che l'utente visualizza nella UI[3] sono derivati dal modello.
- **view**: è l'interfaccia utente, ciò che l'utente vede e con il quale interagisce; è dinamica ed è generata in base al modello dell'applicazione.
- **controller**: è costituito dalla business logic e dal livello presentazionale; esso si occupa del recupero dei dati e decide come presentare il modello, quali parti di esso mostrare.

In conclusione i concetti chiave sono due:

- **Single Respondability Principle**, ogni unità ha una ed una sola responsabilità: il modello rappresenta i dati, la view costituisce la UI e il controller si occupa della logica applicativa.
- **Unità il più possibile indipendenti tra di loro**: questo rende il codice più modulare, riusabile e manutenibile.

4.4 Alcuni concetti chiave della filosofia di AngularJS

Alcuni principi alla base del funzionamento di AngularJS che rendono più semplice ed efficiente lo sviluppo delle applicazioni.

Data-driven (attraverso il *two-way data-binding*)

In un'applicazione Angular non è necessario scrivere grandi quantità di codice per tenere costantemente aggiornate le informazioni visualizzate nella view (provenienti dal model); basta stabilire un binding su un'informazione mostrata nella UI ed essa verrà aggiornata automaticamente al cambiamento del modello sottostante; è sufficiente usare le cosiddette *Expressions* (discusse

[3]User Interface

più avanti). Ma il meccanismo di binding di Angular è ancora più evoluto: il *two-way data-binding* assicura che il controller e la view condividano lo stesso model, quindi nel caso in cui le modifiche arrivino dalla view, ad esempio attraverso l'inserimento di informazioni in un form o interagendo in qualche altro modo con l'interfaccia, il controller avrà accesso a queste informazioni aggiornate costantemente.

Dichiarativo

AngularJS promuove il paradigma dichiarativo nel quale lo sviluppatore dichiara direttamente nel codice HTML ciò che vuole realizzare; ciò è reso possibile dalle *direttive* (trattate più avanti) le quali, sostanzialmente, permettono di estendere il vocabolario di HTML, che si tratti di nuovi elementi o di attributi, per aggiungere delle nuove funzionalità personalizzate. Questo porta a 2 principali vantaggi:

- osservando il codice HTML si può intuire non solo quale sia la struttura dell'applicazione, ma anche cosa rappresentino i singoli elementi
- il comportamento, gli algoritmi, la business logic di ogni componente sono incapsulati nelle direttive create dallo sviluppatore

Separazione delle competenze

Riprendendo quanto detto relativamente al pattern MVC, abbiamo 3 principali componenti nell'applicazione ed ognuna ha delle competenze diverse:

- **model**: i dati da mostrare, spesso rappresentati usando oggetti JSON
- **view**: la pagina HTML che l'utente vede e con la quale interagisce
- **controller**: codice javascript che si occupa della logica applicativa

Dependency Injection

Dependency Injection è un *design pattern* utilizzato per la gestione delle dipendenze tra i vari moduli: un controller o un servizio (trattati più avanti) richiedono una determinata dipendenza (un altro controller o servizio) e questa gli viene fornita (a run-time) da un particolare oggetto, l'*injector*; il controller (o servizio) che richiede una dipendenza può quindi evitare di dover istanziare l'oggetto richiesto o richiederlo invocando una funzione.

4.5 Le componenti principali di un'applicazione Angular

Verrà ora fornita una rapida panoramica delle componenti essenziali presenti in un'applicazione che sfrutti appieno le potenzialità di un framework come AngularJS. Tutte ciò che verrà descritto è alla base della struttura dell'applicazione sviluppata.

4.5.1 Expressions

Una *Expression* di Angular è molto simile ad una espressione in javascript[4]: può contenere stringhe, operatori e variabili. Le expressions sono riconoscibili nel codice HTML in quanto sono poste all'interno di doppie parentesi graffe, ad esempio: "{{ 10 + 3 }}", "{{controller.variable}}". L'espressione presente all'interno delle parentsi verrà valutata ed il risultato verrà mostrato nella UI; le expressions sono il principale meccanismo per il *one-way data-binding*: se all'interno dell'espressione è presente una variabile, il suo valore sarà tenuto costantemente aggiornato, se esso cambia nel model, le modifiche saranno visibili anche nella UI.
In definitiva, le expressions consentono di inserire valori dinamici nel codice html, cioè nella view che sarà mostrata all'utente.

[4]per ulteriori dettagli: https://docs.angularjs.org/guide/expression

4.5.2 Moduli

I moduli in Angular definiscono le applicazioni; può essere considerato come un pacchetto che contiene diverse parti dell'applicazione: controllers, direttive, filtri, servizi, etc..
Sostanzialmente, definisce delle funzioni e le rende accessibili ad ogni componente al suo interno.
Esso può anche avere delle dipendenze da altri moduli, dipendenze definite quando il modulo è istanziato; ciò permette ad un modulo di accedere alle funzioni, controllers, servizi, filtri e direttive di un altro modulo. Nello sviluppo di un'applicazione è necessario definire un modulo che Angular utilizzerà come *entry point* nella fase di *bootstrap* dell'applicazione: ciò si realizza utilizzando la direttiva *ng-app* (discussa più avanti).

4.5.3 Scope

Uno scope in Angular indica il contesto in cui vengono gestiti i dati dell'applicazione (model) e vengono valutate le espressioni; è un semplice oggetto javascript al quale si possono aggiungere proprietà che rappresentano variabili o funzioni. Gli scopes sono organizzati in una struttura gerarchica che segue la struttura del DOM dell'applicazione: c'è un solo *root scope* alla radice della gerarchia, esso gestisce le variabili globali; ci sono poi altri scopes "figli" del root scope,ognuno dei quali è legato ad un controller. Lo scope è l'oggetto collante tra un controller e la relativa view.

4.5.4 Controllers

I controllers in Angular sono regolari oggetti javascript che, come suggerito dal nome, controllano i dati di un'applicazione.
Un controller aggiunge funzionalità allo scope di una view e si occupa dell'esecuzione della maggior parte delle operazioni *UI-oriented*, definisce il comportamento dell'applicazione; alcune tipiche responsabilità di un controller sono:

- recuperare le informazioni da mostrare in una certa parte della UI, settando lo stato di certe proprietà dello scope di riferimento
- decidere quali di queste informazioni mostrare
- presentation logic: come mostrare gli elementi
- user interaction: come gestire gli eventi scatenati dall'interazione dell'utente con elementi della view, ad esempio, cosa fare quando l'utente clicca su un certo bottone

Un controller è sempre legato ad una view, se un controller non è usato nella UI, semplicemente non ha senso di esistere: queste componenti fungono da ponte tra il model, che rappresenta i dati, e le view, che è ciò che l'utente vede.
In generale, queste componenti non dovrebbero svolgere troppe attività, dovrebbero occuparsi solo della business logic relativa ad una singola view; se in un controller vengono gestite operazioni e variabili non strettamente legate ad una view, molto probabilmente sarebbe meglio utilizzare un servizio per incapsulare determinate funzionalità.

4.5.5 Direttive

Una direttiva in Angular è una struttura di markup (elemento o attributo) che permette di estendere HTML con componenti personalizzate, nello specifico: una direttiva è un "marker" in un elemento del DOM che segnala al compilatore HTML di AngualrJS ($compile) di fornire un determinato comportamento a tale elemento. Ci sono due principali tipologie di direttiva:

- **modificatori di comportamento**: questo tipo di direttive funzionano su un elemento html già presente e aggiungono o modificano il comportamento della UI; un esempio di tale direttiva è "ng-show". Esse sono riconoscibili dal prefisso "ng-".

- **componenti riusabili**: una direttiva che crea una nuova struttura html e le fornisce una logica di rendering (come e cosa deve mostrare) e una logica di business (il comportamento); esempi di tali direttive sono i widget, cioè componenti UI riusabili, fornite ad esempio da librerie esterne. Le direttive create per lo sviluppo dell'applicazione BEX sono prevalentemente di questa tipologia.

Il framework offre un insieme di direttive facilmente utilizzabili; verranno ora descritte brevemente le principali direttive Angular utilizzate per lo sviluppo di BEX:

- **ngApp**: è la principale e più importante direttiva, identifica la sezione di HTML controllata da AngularJS e il modulo da utilizzare come entry point al bootstrap dell'applicazione

- **ngBind**: sostituisce il testo all'interno di un elemento html con il valore di un'espressione e lo aggiorna quando tale valore cambia

- **ngController**: istanzia e fissa un controller ad una view; questa direttiva rappresenta uno dei punti chiave nella realizzazione del pattern architetturale MVC

- **ngModel**: una delle principali direttive per la realizzazione del *two-way data-binding*; permette di effettuare il binding tra un elemento di form control (ad esempio, input, select o textarea) ed una property dello scope del controller

- **ngRepeat**: istanzia un certo template per ogni elemento all'interno di una collezione ed ogni istanza di questo template ha un suo scope; questa è una delle direttive più utilizzate nell'applicazione

- **ngClass**: consente di impostare (aggiungere o rimuovere) dinamicamente delle classi CSS su un certo elemento; con ng-class si possono usare sia stringe che oggetti come valori

- **ngShow**: con questa direttiva è possibile visualizzare o nascondere un elemento in base alla "truthiness" del valore di una variabile

- **ngSwitch**: permette di mostrare selettivamente determinati elementi all'interno di un contenitore, in base al valore di una variabile; la logica di funzionamento è simile allo *switch* nei linguaggi di programmazione

- **ngClick**: consente di specificare una funzione (o anche direttamente del codice) da invocare (o eseguire) quando si verifica l'evento di *click* su un certo elemento

- **ngMouseover**: simile ad ngClick, ma l'evento di interesse è il *mouse over* su un elemento.

- **ngValue**: effettua il binding tra un'espressione e il valore di un elemento *option* o *input[radio]*, in modo tale che quando questo elemento è selezionato, il *ngModel* di quell'elemento è impostato al valore su cui è stato effettuato il binding.

- **ngInit**: permette di inizializzare una variabile assegnandole un valore iniziale.

4.5.6 Servizi

In un'applicazione Angular, i servizi sono degli oggetti o delle funzioni che possono mantenere il loro stato in tutta l'applicazione, sono istanziati una sola volta (sono dei *singleton*), quindi ogni oggetto (controller o altri servizi) che ne richieda l'utilizzo, accede alla stessa istanza; questa è la principale differenza dai controller i quali invece vengono distrutti e re-istanziati più volte durante l'utilizzo dell'applicazione; inoltre i controllers non possono comunicare direttamente tra di loro per condividere comportamenti e stati: questa è una delle principali utilità di un servizio, permettere la comunicazione tra più controllers.
Angular mette a disposizione alcuni fondamentali servizi e rende possibile

implementarne di propri; per lo sviluppo di BEX sono stati implementati svariati servizi (descritti nella sezione 4.6.2) e ne sono stati utilizzati alcuni forniti dal framework, il principale di questi è $http.

Servizio $http

Il servizio $http di AngualrJS è utilizzato per facilitare la comunicazione con un server remoto HTTP, cioè fare richieste GET e POST e gestire le risposte: sostanzialmente permette di effettuare chiamate asincrone ad un server. Trattandosi di richieste asincrone le risposte potrebbero arrivare dopo un tempo indefinito, il servizio permette quindi di gestire tale situazione utilizzando una particolare interfaccia: la *promise interface*.
Ogni chiamata asincrona, ad esempio un'invocazione di "$http.get()", ritorna un promise object; questo promise object incapsula al suo interno due funzioni: una funzione *success* e una funzione *error*.
Queste sono due funzioni di callback; quando arriverà la risposta dal server verrà eseguita una di queste due funzioni, in base al tipo di risposta (successo o fallimento). Il parametro passato a queste funzioni è un oggetto che rappresenta la risposta del server, ha property relative a dati, stato della risposta e header. Questo meccanismo di esecuzione di callback per la gestione delle chiamate asincrone permette di non bloccare tutta l'interfaccia nell'attesa che arrivi una risposta dal server, consentendo all'utente di poter continuare nella navigazione all'interno dell'applicazione.

4.5.7 Filtri

Un'altra fondamentale funzionalità fornita da AngularJS sono i filtri: essi permettono di processare dati e formattare i valori da presentare all'utente, astraendo la visualizzazione dalla reale forma con cui i dati sono memorizzati.
Quando vengono applicati dei filtri a dei dati, cambia la visualizzazione fornita all'utente, ma i valori reali non vengono modificati.
Per applicare i filtri si usa la sintassi "{{expression | filter }}": il filtro

prenderà il valore dell'espressione e lo convertirà in qualche altra forma in base alla tipologia di filtro. Qui di seguito vengono brevemente descritti i principali filtri usati nell'applicazione.

- **orderBy**: ordina uno specifico array in base ad un certo predicato passato come parametro; accetta inoltre un secondo parametro opzionale, cioè un boolean per stabilire se l'ordinamento debba essere crescente o decrescente. Questo filtro è stato usato con la direttiva ng-repeat.

- **filter**: utilizzato per filtrare dinamicamente un array in base ad un predicato o ad una funzione, permettendo di decidere quali elementi siano inclusi e quali esclusi. Anche questo filtro è stato usato in combinazione con la direttiva ng-repeat.

Con AngularJS è inoltre possibile definire propri filtri in modo da filtrare un array in base ad una funzione definita dallo sviluppatore; per gli scopi dell'applicazione questa funzionalità è risultata molto utile in quanto ha permesso di implementare filtri e ordinamenti sulla bibliografia, illustrati nella sezione 4.6.4.

4.6 Struttura generale dell'applicazione

In Figura 4.1 è possibile osservare uno schema generale che rappresenta le principali componenti e le relazioni e dipendenze che le legano. In questa sezione verranno descritte queste componenti per fornire un quadro più chiaro di come sia strutturata l'applicazione.

4.6.1 Servizi remoti

Per recuperare le informazioni relative agli articoli scientifici vengono usati due servizi remoti diversi, ognuno con un preciso scopo.

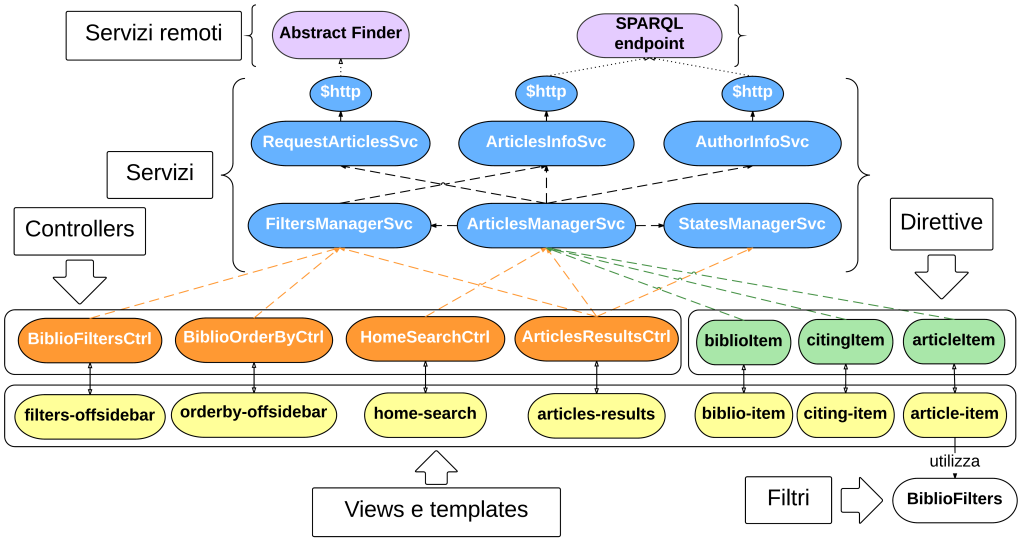

Figura 4.1: Principali componenti dell'applicazione

Abstract finder

Un servizio remoto[5] che permette di ottenere un primo livello di informazioni su un insieme di articoli scientifici ritenuti rilevanti in base a dei termini di ricerca.
Attraverso delle semplici richieste GET si contatta il servizio, passando nei parametri della richiesta una query: questa query è costituita da una stringa e rappresenta i termini per la ricerca di articoli rilevanti.
Il servizio risponderà con un oggetto in formato JSON, il quale tra le sue property contiene una lista con gli URI di livello work degli articoli che verranno mostrati come risultati di ricerca.

SPARQL endpoint

Questo servizio[6] permette di accedere a tutti i dati presenti nel dataset di Semantic Lancet; in particolare, è possibile accedere alle informazioni relative a tutti gli articoli scientifici del triplestore RDF. Per ottenere tali informazioni

[5]http://www.semanticlancet.eu/abstractfinder/
[6]http://two.eelst.cs.unibo.it:8181/sparql.tpl

è necessario effettuare delle query SPARQL al servizio disponibile tramite REST API. Questo sparql endpoint viene contattato prevalentemente per avere tutte le informazioni relative agli articoli dei risultati di ricerca; con una prima interrogazione all'abstract finder si prendono solo gli URI dei work degli articoli ed in seguito, utilizzando questi URI, vengono richieste tutte le ulteriori informazioni sui singoli articoli, ad esempio: titolo, anno di pubblicazione, autori, abstract, citazioni, articoli citanti, etc...

4.6.2 Servizi dell'applicazione

Per lo sviluppo dell'applicazione sono stati implementati diversi servizi Angular, ognuno con un preciso scopo e con determinate responsabilità. Ognuno di questi servizi è stato progettato per seguire determinati criteri:

- essere riusabile: rendere possibile ai diversi controllers l'accesso a dati condivisi
- mantenere stati e variabili non direttamente collegati alle views
- rendere l'applicazione più modulare e facilmente estendibile

Ogni servizio ha una singola responsabilità, si occupa della gestione di un certo dominio di dati o dell'esecuzione di determinate operazioni; infine espone all'esterno (ai livelli sottostanti) un'interfaccia di utilizzo. Come è possibile osservare nella Figura 4.1, i servizi sono divisi su due livelli:

- **1° livello**: **RequestArticlesService, ArticlesInfoService, AuthorInfoService**. I servizi di questo livello hanno come principale obiettivo quello di effettuare le richieste ai server, non si occupano della gestione della risposta ma delegano tale compito al livello di servizi sottostante.
- **2° livello**: **FiltersManagerService, ArticleManagerService, StatesManagerService**. A questo livello vengono materialmente gestiti

i dati provenienti dai server, vengono memorizzate le informazioni sugli articoli, tutto ciò che li riguarda e vengono esposte le funzioni che permettono al sottostante livello di controllers e direttive di ottenere tali informazioni per visualizzarle nella UI.

Tutti i servizi che si occupano di recuperare informazioni contattando server remoti, lo fanno per mezzo di richieste GET: viene utilizzato il servizio $http; ogni funzione che fa uso di questo servizio ritorna l'oggetto *promise* al chiamante (controller o altro servizio) così da rendere possibile la gestione della risposta.

RequestArticlesService

Lo scopo di questo servizio è contattare il servizio Abstract Finder per ottenere la lista degli articoli dei risultati di ricerca.
Rende quindi accessibili le funzioni per impostare i termini di ricerca (la stringa inserita nella casella di ricerca nella home page) e per effettuare materialmente le richieste.

ArticlesInfoService

Anche questo servizio si occupa di contattare un servizio remoto, in questo caso si tratta dello SPARQL endpoint: l'obiettivo è quello di recuperare tutte le informazioni relative agli articoli scientifici. Espone un grande numero di funzioni ognuna delle quali effettua una query all'endpoint sparql per ottenere un certo tipo di informazioni relative ad un articolo, ad esempio: informazioni generiche (titolo, anno, abstract, etc...), autori, info citazionali, bibliografia, articoli citanti, etc...

AuthorInfoService

Un altro servizio il cui fine è quello di recuperare informazioni da un servizio remoto.
AuthorInfoService è responsabile del recupero di dati relativi ad un autore,

ad esempio: tutti gli articoli scritti da un certo autore. Anche in questo caso tali dati vengono ottenuti effettuando query sparql sull'endpoint.

FiltersManagerService

Servizio che ha il compito di gestire i filtri: ha tutte le variabili relative ad ogni filtro, stabilisce i valori di default di queste variabili e le rende accessibili per mezzo di funzioni *getter* e *setter* dedicate.
Questo servizio semplifica la comunicazione tra il controller che "legge" i valori dei filtri, cioè articlesResultsController, e il controller che li imposta, cioè BiblioFiltersController.

ArticlesManagerService

All'interno dell'applicazione, questo è il servizio più importante: si occupa della gestione degli articoli e delle loro informazioni recuperate dai server utilizzando altri servizi.
Qui è presente la collezione degli articoli dei risultati di ricerca: ogni controller o servizio che abbia bisogno di accedere alle informazioni relative ad uno o più articoli, deve interrogare ArticlesManagerService. Svolge un ruolo fondamentale nell'applicazione in quanto incapsula al suo interno tutti i dettagli implementativi sulla gestione dei dati, fornendo all'esterno una semplice interfaccia che assicura un alto livello di astrazione: gli altri controllers e servizi non si devono preoccupare dei dettagli relativi a come le informazioni vengono recuperate, gestite e memorizzate, ma invocano delle semplici funzioni "getter" che nascondono tutta la complessità. In pratica, quando un controller richiede informazioni su un articolo, viene contattato questo servizio il quale delega ad altri servizi (come ArticlesInfoService e RequestArticlesService) le operazioni di interrogazione ai server e gestisce in seguito le risposte che provengono da questi, andando ad arricchire e strutturare la collezione di articoli.

StatesManagerService

Questo è un servizio di supporto alla navigazione; è utilizzato per la gestione delle liste di articoli visualizzate dall'utente nel corso della navigazione. Quando l'utente effettua un nuovo passo nella navigazione all'interno dell'applicazione, viene invocata una funzione di questo servizio per salvare la lista dei risultati precedentemente visualizzati.
Nel momento in cui l'utente "torna indietro" nella navigazione (usando il breadcrumb), allora viene recuperata una determinata lista di articoli.

4.6.3 Views, controllers e direttive

Le views, i controllers a loro legati e le direttive (con le loro funzioni) vanno a costituire il livello più alto dell'applicazione: qui ci sono le interfacce che gli utenti vedono e tutti i comportamenti che rendono possibile le interazioni con i vari elementi visualizzati.
Qui di seguito vengono sinteticamente descritte le componenti principali.

Homepage di ricerca

La view *home-search* e il relativo controller *HomeSearchController* si occupano della homepage visualizzata al caricamento dell'applicazione e realizzano le funzionalità di ricerca "By abstract" e "By title".

Risultati di ricerca

La view *articles-results* è quella visualizzata dall'utente dopo aver effettuato una ricerca; in sè è molto semplice in quanto funge sostanzialmente da contenitore per gli elementi dei risultati di ricerca. Il controller *ArticlesResultsController*, tramite *ArticlesManagerService*, recupera e mantiene aggiornati gli articoli da mostrare e, attraverso il servizio *FiltersManagerService*, ottiene le impostazioni dei filtri; queste informazioni verranno poi passate agli elementi che rappresentano i singoli articoli. Esso si occupa

inoltre di effettuare alcuni controlli e, se necessario, mostra delle notifiche all'utente.

Filtri e ordinamenti

filters-offsidebar (controller: *BiblioFiltersController*) e *orderby-offsidebar* (controller: *BiblioOrderByController*) vanno a comporre gli elementi visualizzabili nella offsidebar dei filtri. I controllers relativi, usando il servizio *FiltersManagerService* recuperano le impostazioni di default dei filtri e degli ordinamenti e permettono di modificarne il valore, in base alle interazioni dell'utente con gli elementi presenti nella offsidebar.

Articolo dei risultati

ArticleItem è una delle tre principali direttive implementate e, con il template *article-item*, va a costituire il widget che rappresenta il singolo articolo nei risultati di ricerca.
Sorvolando sugli aspetti presentazionali (discussi precedentemente), è interessante fare alcune osservazioni sulla direttiva:

- la direttiva è stata implementata come un nuovo elemento HTML
- tutta la business logic è contenuta nella *link function* e le principali funzioni sono relative all'interazione dell'utente con gli elementi visualizzati
- scope isolato: lo scope della direttiva non eredita nulla dallo scope del *parent* e ogni dato che questo *parent* ha necessità di passare alla direttiva, deve essere passato attraverso degli attribuiti appositamente definiti. Questa è la soluzione migliore per rendere riusabile la componente, rendendola indipendente dal contesto in cui è usata

Le informazioni passate alla direttiva attraverso gli attributi sono le informazioni relative all'articolo e ai filtri (e ordinamenti) da applicare alla bibliografia; questi ultimi dati sui filtri verranno usati per l'effettiva attività di *filtering* sulla bibliografia, verranno infatti applicati ai singoli elementi *biblio-item*.

Elemento della bibliografia

biblioItem è un'altra delle principali direttive create durante lo sviluppo e, con il template *biblioItem*, costituisce il singolo elemento nella bibliografia di un articolo (sezione "*Cited articles*"). Anche in questo caso si possono fare le medesime osservazioni fatte per *articleItem* relativamente alla tipologia di direttiva e allo scope isolato. Una nota da aggiungere per questa direttiva: nella sua *link-function* ci sono anche le funzioni relative al parsing dei dati da visualizzare nel donut chart dei motivi citazionali e le impostazioni per il rendering dello stesso.

Articolo citante

citingItem è un'ulteriore direttiva, sviluppata per la rappresentazione degli articoli che citano un altro articolo; questi elementi vengono visualizzati nella sezione "Cited by" della finestra di dialogo relativa agli articoli citanti. Anche in questo caso valgono le osservazioni fatte per le precedenti direttive.

4.6.4 Filtri implementati

Come detto nelle precedenti sezioni, per consentire le attività di *filtering* sugli elementi della bibliografia, oltre ai filtri *built-in* di AngularJS, ne sono stati implementati altri "personalizzati"; questo si è reso necessario in quanto i filtri offerti dal framework non si sono rivelati abbastanza flessibili per adattarsi alle esigenze dei filtri che si sarebbero dovuti implementare.
In *BiblioFilters* sono stati implementati tutti i filtri utilizzati per la bibliografia. Seppure il codice dei singoli filtri non sia lo stesso, la logica implementata è sostanzialmente la stessa: ad ogni filtro vengono passati due parametri, il primo è l'array con tutti gli elementi da filtrare, il secondo è una condizione che andrà a stabilire quali elementi saranno inclusi nei risultati filtrati e quali saranno esclusi, infine viene ritornata la lista con gli elementi filtrati.
I filtri implementati sono i seguenti:

- **afterYear**: nella lista filtrata ci saranno solo gli articoli con anno di pubblicazione maggiore o uguale ad un determinato anno
- **onlySelfcitation**: nella lista filtrata ci saranno solo gli articoli identificati come self-citation (auto-citazione)
- **characterizations**: nella lista filtrata ci saranno solo gli articoli citati per certo insieme di motivi
- **authors**: nella lista filtrata ci saranno solo gli articoli scritti da certi autori

4.7 Altri frameworks, librerie, template e strumenti utilizzati per lo sviluppo

Oltre al framework AngularJS, per lo sviluppo dell'applicazione sono stati utilizzati anche altri frameworks, librerie e svariati tool di assistenza all'esecuzione di determinati task ripetitivi; qui di seguito ne viene fornita una breve rassegna.

Bootstrap e templates

Per lo sviluppo di BEX e del design della sua interfaccia web è stato ampiamente utilizzato un framework HTML, CSS e javascript: Bootstrap[7].
Bootstrap è costituito da un insieme di elementi grafici, stilistici, di impaginazione e javascript pronti all'uso; esso include infatti template HTML e CSS per la tipografia, per i bottoni, finestre modali e altri elementi del genere. Inoltre Bootstrap dà la possibilità di realizzare layout *responsive*, ciò sostanzialmente permette di cambiare il posizionamento e le dimensioni degli elementi grafici visualizzati nell'interfaccia in modo da adattarsi automaticamente alle dimensioni dello schermo: si rende l'esperienza d'uso più fluida anche su schermi di dimensioni ridotte (tablet e smartphone) riducendo la

[7]http://getbootstrap.com/

necessità di effettuare zoom sugli elementi o dover scorrere orizzontalmente per visualizzare i contenuti della pagina. Per lo sviluppo si è scelto di partire da una soluzione di templating che fornisse anche una struttura base per l'applicazione come punto di partenza per le successive fasi di progettazione; si optato allora per un template Bootstrap che fornisse anche una buona integrazione di partenza con il framework AngularJS: è stato scelto il template "Angle" fornito da Themicon[8]; di questo template sono state prese le componenti ritenute più utili per poi riadattarle e personalizzarle per gli scopi progettuali di BEX.

ngDialog

Per Tutte le finestre di dialogo e le finestre modali utilizzate nell'applicazione è stata usata la libreria javascript "ngDialog"[9] che garantisce un'ottima integrazione con AngularJs.

CanvasJs

Per la realizzazione di tutti i grafici visualizzabili navigando l'applicazione è stata usata una libreria di *charting* che utilizza javascript e le canvas di HTML5: CanvasJS[10]. Essa fornisce delle API semplici da utilizzare e garantisce un'ottima compatibilità su più browser e dispositivi.

BootstrapUI

BootstrapUI[11] è una libreria che mette a disposizione una serie di componenti tipici di Bootstrap ma forniti sotto forma di direttive Angular: questo ha permesso di integrare nell'applicazione delle componenti in modo più efficiente senza dover utilizzare jQuery[12].

[8] https://wrapbootstrap.com/user/themicon
[9] https://github.com/likeastore/ngDialog
[10] http://canvasjs.com/
[11] http://angular-ui.github.io/bootstrap/
[12] http://jquery.com/

AngularUI Router

BEX è una single page application e nel passaggio da una view all'altra non viene effettuata una nuova richiesta al server per ricaricare l'intera pagina, ma vengono richieste solo le risorse informative da visualizzare nella nuova view; per realizzare questo passaggio tra le view viene utilizzato un particolare meccanismo, il *routing*. Per l'implementazione del *routing* in BEX è stata utlizzata la libreria AngularUI Router[13]; il suo utilizzo si basa sul concetto di "stato" che corrisponde ad un qualsiasi "luogo" nell'applicazione, in termini di UI e navigazione. Dopo aver definito uno stato è possibile stabilire come l'applicazione debba comportarsi quando viene "incontrato" un certo stato. Questa libreria è stata usata perchè permette di rendere il codice più modulare e facilmente manutenibile.

Npm, Bower, Gulp

Per l'installazione dei vari pacchetti e moduli javascript utilizzati per la realizzazione di BEX, è stato usato il *package manager npm*[14].
Npm è stato usato per installare diversi moduli, due in particolare sono degni di nota.
Bower[15]: un altro *package manager* particolarmente utile per l'installazione di librerie utilizzate per lo sviluppo front-end; Bower si occupa inoltre di gestire automaticamente tutte le dipendenze e di tenere traccia di queste ultime nel file *bower.json*.
Un altro modulo che si è rivelato molto utile è stato *Gulp*[16]: un *task manager* che permette di automatizzare l'esecuzione di determinati task ripetitivi come ad esempio la concatenazione e la *minification* di files javascript.

[13]https://github.com/angular-ui/ui-router
[14]https://www.npmjs.com/
[15]http://bower.io/
[16]http://gulpjs.com/

Server-side

Per le attuali funzionalità dell'applicazione non sono stati necessari particolari script server-side, l'unica funzione del server (attualmente) è quella di rispondere alle richieste dei client per il caricamento dell'intera applicazione; tutte le successive richieste sono indirizzate ad altri server (abstract finder e sparql endpoint).
Server-side è stato utilizzato il framework javascript Node.js[17] ed in particolare è stato usato un package npm: *http-server*[18].
http-server è un pacchetto per node.js che permette di configurare un semplice server http da riga di comando: è abbastanza prestante da poter essere usato in *production* ma abbastanza flessibile e configurabile per essere utilizzato anche in fase di sviluppo e *testing*; dopo aver eseguito diverse prove, questo strumento si è rivelato abbastanza robusto e stabile, si è quindi deciso di utilizzarlo per la prima versione dell'applicazione. Non si esclude un futuro passaggio a strumenti più evoluti nel caso in cui si renda necessario eseguire delle attività di scripting server-side più elaborate.

[17] http://nodejs.org/
[18] https://www.npmjs.com/package/http-server

Capitolo 5

Valutazione

5.1 Valutazioni sul recupero delle informazioni

In seguito ad una ricerca effettuata dall'utente, come già detto, viene contattato il servizio "abstract finder" per ottenere gli URI di livello Work degli articoli da mostrare nei risultati di ricerca; dopo aver ottenuto queste risorse è necessario recuperare tutte le informazioni relative ai singoli articoli, ciò viene effettuato con un certo numero di chiamate asincrone, specifiche per ognuno dei risultati; in questo modo l'applicazione risulta più reattiva e il singolo articolo viene visualizzato nella view non appena arrivano dal server le informazioni su quel determinato contenuto.
Inoltre, si è pensato di suddividere su più livelli di dettaglio le informazioni che riguardano il singolo articolo:

1. **Informazioni generiche**: titolo, anno di pubblicazione, autori, volume, issue, link esterno, abstract

2. **Informazioni aggregate sulle citazioni ricevute**: numero di citazioni globali, numero di atti citazionali, numero di articoli citanti; da sottolineare che questi sono solo dati numerici aggregati, non ci sono ulteriori dettagli

3. **dettagli sulle citazioni ricevute**: il dettaglio su tutte le citazioni ricevute, motivi citazionali, contesti citazionali, articoli citanti

4. **elementi della bibliografia**: tutte le informazioni sugli articoli che compongono la bibliografia

Questa soluzione consente di ottenere due principali vantaggi.

Maggiore efficienza e minore impatto sul server

Per recuperare tutte le possibili informazioni relative ad un articolo (tutti i livelli) può essere necessario effettuare un numero considerevole di richieste al server (SPARQL endpoint), dipende molto dal numero di articoli citanti e di elementi della bibliografia; se si considera che queste richieste vanno eseguite per ogni risultato di ricerca ci si rende subito conto che il numero di interrogazioni al server potrebbe diventare decisamente elevato.
Inoltre, queste richieste verrebbero fatte tutte in un lasso di tempo molto breve, nell'ordine dei pochi decimi di secondo: un carico sul server che potrebbe risultare anche non sostenibile. Ci sarebbero anche delle ripercussioni sull'interattività dell'applicazione, la quale, trovandosi a dover gestire una grande mole di dati in ingresso, potrebbe andare incontro a dei momentanei *freeze*, andando ad intaccare l'esperienza utente.
Riflettendo sul tipo di attività che l'utente esegue utilizzando l'applicazione, si è giunti alla conclusione che effettuare tutte queste richieste è, semplicemente, non necessario. L'utente, dopo aver effettuato una ricerca, ha come primo obiettivo quello di farsi una prima idea sui risultati; è infatti abbastanza improbabile che egli voglia visualizzare immediatamente tutte le informazioni su tutti gli articoli o, perlomeno, non sono stati presi in considerazione tasks che richiedano un'operazione del genere; egli molto probabilmente sarà interessato ad un ristretto insieme degli articoli dei risultati.
Per compiere una prima selezione degli articoli ritenuti più interessanti è sufficiente fornire alcune informazioni essenziali, quali il titolo, l'anno di pubblicazione e gli autori: questi, infatti, sono i dati mostrati nell'intestazione

di ogni risultato di ricerca (in realtà è presente anche il link all'articolo nella Digital Library, utile per avere un collegamento rapido).
Quando l'utente vuole approfondire lo studio di un articolo ritenuto interessante, è sufficiente cliccare sull'apposito bottone nella parte bassa del pannello per avere ulteriori informazioni; è proprio questo il punto fondamentale: nel momento in cui l'utente clicca su questo bottone, egli si dichiara interessato a tale articolo, ed è dunque sensato recuperare tutti gli ulteriori dettagli informativi affinchè egli possa analizzarli.
Quindi, la prima conclusione è la seguente: non è necessario richiedere subito tutte le informazioni su tutti gli articoli in quanto l'utente difficilmente si ritroverà ad analizzarli tutti nel dettaglio; è quindi più logico richiedere i dettagli su un certo articolo solo quando l'utente si dimostra effettivamente interessato ad approfondirne l'analisi. Ma in realtà le informazioni che vengono visualizzate al primo *drill-down* su un articolo sono già state caricate insieme alle informazioni generiche dello stesso: si tratta dell'abstract e delle informazioni aggregate sulle citazioni in entrata. L'utente non dovrà attendere che tali informazioni vengano recuperate dal server. Ciò che viene realmente richiesto (al server) al *drill-down* è costituito dalle informazioni di livello 3 (dettagli sulle citazioni ricevute) e 4 (elementi della bibliografia): informazioni per la cui visualizzazione è necessaria un'ulteriore interazione da parte dell'utente, esse infatti non sono mostrate immediatamente in quanto potrebbero anche non essere rilevanti per il task che si sta eseguendo. Sostanzialmente, questi dettagli vengono caricati in background (informazioni che comunque necessitano, sperabilmente, di pochi decimi di secondo per essere recuperate dal server) mentre l'utente è ancora impegnato nella lettura dell'abstract o dei dati aggregati sulle citazioni in entrata.
Riassumendo, al caricamento dei risultati di ricerca, per ogni articolo vengono richieste immediatamente le informazioni di livello 1 e 2, esse rappresentano le informazioni essenziali per una prima valutazione e sono recuperabili con poche e veloci query allo sparql endpoint; quando l'utente si dichiara interessato, vengono richieste le informazioni di livello 3 e 4, dettagli sulle citazioni

e sulla bibliografia il cui recupero prevede un numero non determinabile a priori di query più complesse e potenzialmente più lente da eseguire per lo sparql endpoint.
In definitiva, vengono effettuate meno richieste ed esse sono "diluite" in un lasso di tempo ampio abbastanza da non appesantire di lavoro il server; all'utente tutto ciò è totalmente trasparente in quanto nella stragrande maggioranza dei casi non percepirà alcun ritardo nel caricamento delle informazioni sul singolo articolo, anzi, l'adozione di questa soluzione permetterà un caricamento decisamente più veloce dei risultati di ricerca.

Migliore suddivisione delle responsabilità

Per ognuna di queste categorie di informazioni sono previste diverse funzioni, ognuna delle quali ha la responsabilità di recuperare, memorizzare o rendere accessibili le informazioni di una specifica categoria. Non ci sono funzioni che si occupano di gestire informazioni eterogenee.
Questa suddivisione dei compiti ha contribuito a rendere più modulare la struttura dell'applicazione.

5.2 Test con gli utenti

Una volta realizzata l'applicazione è stata effettuata una prima ispezione del risultato da parte del team che ha seguito lo sviluppo del progetto: il servizio è stato ritenuto valido sotto tutti i punti di vista in quanto è risultato in linea con i requisiti stabiliti nella prima fase di analisi e progettazione.
In seguito al rilascio dell'applicazione si è deciso, in collaborazione con un gruppo di ricercatori, di ideare subito anche un primo test da proporre agli utenti finali in modo da poter raccogliere dei primi *feedbacks* utili. Al momento della scrittura di questa trattazione il test realizzato non rappresenta una versione definitiva ed è, esso stesso, sottoposto a valutazione da parte degli utenti che lo eseguono.
Il numero di risposte attualmente ricevute non è ancora abbastanza alto da

poter dare valenza statistica significativa ai risultati; perciò in questa fase ci si limiterà a delle brevi considerazioni sulle tendenze che stanno emergendo dalle risposte.

Struttura del test

Il test ha lo scopo di valutare l'usabilità e l'efficacia generale di BEX nel supportare l'utente nell'esecuzione di alcuni semplici tasks riconducibili a quelli trattati nei precedenti capitoli. Esso per la sua esecuzione richiede circa 45 minuti ed è suddiviso in più parti.

Background

Vengono proposte delle domande sul background culturale dell'utente e sulla sua familiarità con l'utilizzo delle digital libraries. L'intenzione è quella di identificare meglio la tipologia di utente, comprendere quanto esso sia esperto nell'ambito del Semantic Web e delle digital libraries.

Warm-up

All'utente vengono concessi 5 minuti per prendere familiarità con le funzionalità offerte da BEX, utilizzando, eventualmente, la guida rapida all'uso accessibile direttamente nell'applicazione.

I tasks

Attualmente sono stati proposti dei prototipi di task ognuno dei quali prevede che l'utente utilizzi l'applicazione per cercare informazioni relative ad articoli, bibliografie ed autori per poi rispondere a delle domande relative a tali informazioni. La correttezza delle risposte permetterà di stabilire se l'utente abbia eseguito il task con successo.

- **primo task**: richiede di cercare un determinato articolo (per titolo) e di rispondere ad alcune domande relative ad autocitazioni nella

bibliografia, motivi citazionali ed anno di pubblicazione degli articoli citati

- **secondo task**: richiede di effettuare una ricerca partendo da alcune parole chiave e di prendere in analisi l'articolo (dei risultati) che ha ricevuto più citazioni globali; saranno quindi poste domande relative alle citazioni ricevute da tale articolo

- **terzo task**: dopo aver effettuato una ricerca per titolo viene chiesto all'utente di stabilire il numero di articoli scritti da ogni autore dell'articolo inizialmente cercato

Questionario

Vengono poste le 10 domande del System Usability Scale (SUS)[1]: si tratta di un test con un protocollo fisso ed un criterio standard per la valutazione dei risultati, viene infatti applicato un algoritmo. Le domande sono le seguenti:

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use.

[1] http://www.measuringu.com/sus.php

9. I felt very confident using the system.

10. I needed to learn a lot of things before I could get going with this system

Le 10 domande sono poste alternativamente con enunciazione positiva e negativa e ad esse bisogna dare una risposta secondo una scala di Likert da 1 (*Strongly disagree*) a 5 (*Strongly agree*). Applicando una funzione[2] a tali risposte si ottiene un punteggio compreso tra 0 e 100 il quale è considerabile come un indicatore dell'usabilità del sistema; punteggi superiori al 68 indicano una buona usabilità.

Domande aperte

Vengono proposte alcune domande aperte (non obbligatorie) con le quali si chiede un parere dell'utente sull'esperienza d'uso del sistema, sulle funzionalità ritenute più utili e sui punti deboli dell'applicazione. Viene concessa inoltre la possibilità di suggerire modifiche all'applicazione o nuove funzionalità che potrebbero risultare comode. Il test si conclude con due domande sulla struttura del test stesso e sulla guida all'uso di BEX.

I primi risultati

Tipologia di utente

Dalle risposte finora ricevute è emerso un insieme di utenti abbastanza omogeneo e con caratteristiche simili:

- professori e ricercatori universitari
- hanno un background di conoscenze di Semantic Web e Semantic Publishing
- conoscono le digital libraries e ne fanno un uso frequente

[2] http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

- sono esperti nell'analisi di articoli scientifici e generalmente utilizzano la rete citazionale come principale strumento di navigazione

Alcune considerazioni

Per quanto riguarda i tasks è emerso che, sostanzialmente, non sono stati riscontrati particolari problemi nella loro esecuzione: tutti gli utenti hanno risposto correttamente alla maggior parte delle domande, esclusi piccoli casi di errori che potrebbero essere collegati alla potenziale ambiguità di alcune domande.
Dalle domande aperte proposte al termine del test sono stati raccolti i *feedbacks* riportati qui di seguito.
Funzionalità ritenute più utili:

- accesso veloce alle informazioni sugli articoli della bibliografia e sugli articoli citanti; è stata apprezzata la visualizzazione dei motivi citazionali.
- navigazione tra gli articoli per mezzo della rete citazionale
- filtri sulla bibliografia
- ricerca semplificata

Suggerimenti su funzionalità e mancanze evidenziate:

- rendere più evidente il breadcrumb per la navigazione e integrarlo con i meccanismi di history del browser
- aggiungere più labels informative
- modalità di ricerca nella homepage proposte come tabs e non come bottoni
- difficoltà nel recupero di informazioni relative ad un autore di interesse

In alcuni casi si sono verificati alcuni malfunzionamenti durante l'utilizzo dell'applicazione; su essi si è già indagato:

- problema di caricamento dei risultati per particolari termini di ricerca: dovuto ad un'inaspettata struttura della risposta proveniente dal servizio *abstract finder*. Si sta già provvedendo alla risoluzione del problema

- problemi di visualizzazione dei *citing articles* nella apposita finestra modale: riscontrabile con l'utilizzo dell'applicazione su tablet; il bug, molto probabilmente, è riconducibile ad un conflitto con una delle librerie esterne.

Dalle risposte del questionario SUS è stata calcolata una prima media, ottenendo il seguente punteggio: 83.13/100
Questo punteggio è da ritenersi solamente indicativo in quanto non è ancora stato raggiunto un numero di risposte al test tali da conferirgli piena valenza.

Conclusioni e sviluppi futuri

> "Fate le cose nel modo più semplice possibile, ma senza semplificare"
>
> *Albert Einstein*

In questa trattazione è stata presentata BEX, un'applicazione web nata con il preciso obiettivo di supportare i ricercatori universitari nell'esecuzione di determinati tasks di ricerca e valutazione di articoli scientifici; attività che richiedono l'analisi di una grande quantità di informazioni. Il progetto è nato dal confronto diretto con un gruppo di ricercatori che si occupano quotidianamente dello sviluppo del *Semantic Lancet Project*[3]: esso mette a disposizione un dataset ricco di informazioni su articoli scientifici ma è carente dal punto di vista dei servizi che permettano di fruire in modo semplice ed intuitivo di tali informazioni. Dopo alcuni incontri con questi ricercatori sono emersi alcuni dei principali requisiti funzionali ai quali lo strumento che si andava a sviluppare avrebbe dovuto saper rispondere.
Nella fase di raccolta delle specifiche, l'attenzione si è concentrata soprattutto su alcuni elementi: le bibliografie degli articoli e, più in generale, la rete che si viene a creare tra di essi attraverso le citazioni: una rete nella quale i collegamenti tra i vari nodi non sono semplici archi tra un articolo "A" ed un articolo "B", ma, grazie ai dati presenti nel *Semantic Lancet Triplestore*, vengono rese disponibili ulteriori informazioni preziose, come i motivi citazionali (perchè "A" cita "B") e i contesti citazionali (cosa dice "A" a proposito

[3]http://www.semanticlancet.eu/

di "B").
Fornire questi dati unitamente alle informazioni generiche sul singolo articolo permette al ricercatore di svolgere in modo più rapido le attività necessarie per raggiungere specifici *goals* di analisi e valutazione; alla rapidità si accosta anche l'efficienza nel momento in cui l'applicazione veicola le informazioni in modo tale da facilitare l'attribuzione di senso da parte dell'utente.
Favorire il *sensemaking* costituisce una delle principali chiavi di volta di tutto il progetto: aiutare l'utente nella ricerca di una rappresentazione che gli permetta di mettere in relazione le informazioni; il modo migliore per ottenere ciò consiste nello studiare le esigenze dell'utente e nel ridurre il più possibile il carico cognitivo richiesto.
BEX rappresenta un primo passo in questa direzione: un'applicazione realizzata appositamente per i ricercatori, semplice da usare, con un'interfaccia user-friendly composta dalle sole funzionalità ritenute essenziali per un'esecuzione efficiente delle attività che il ricercatore ha in mente.
Per lo sviluppo di BEX è stata necessaria un'ampia fase di analisi, progettazione e sperimentazione: un percorso che ha visto l'implementazione e la successiva sostituzione di diversi prototipi di funzionalità; una serie di tentativi i quali, di volta in volta, hanno messo in luce i punti di potenziale miglioramento sui quali si è successivamente lavorato.
Uno dei principali aspetti innovativi del servizio realizzato è rappresentato dal connubio di tecnologie eterogenee: alle tecnologie del Semantic Web sono state affiancate le più moderne tecnologie del *web development* (AngularJS principalmente) e del *web design*, creando così un'applicazione sostanzialmente unica nel suo genere.
Il risultato a cui si è giunti rappresenta un primo punto di arrivo: l'applicazione, dopo un'ispezione da parte del team che ha assistito lo sviluppatore, è stata ritenuta completa dal punto di vista funzionale in quanto permette di rispondere a tutti i requisiti inizialmente stabiliti. Ma ciò non vuol dire che lo sviluppo dell'applicazione sia concluso: quanto si è realizzato costituisce la prima fondamentale iterazione in un ciclo di sviluppo che si compone di

più fasi. Si è arrivati ad un prima versione completa che può essere rilasciata e aperta al pubblico; la fase successiva, che può durare anche mesi, consiste nel raccogliere dei *feedbacks* da parte degli utenti finali, un'operazione indispensabile per poter individuare i punti di forza, ma soprattutto i punti sui quali è necessario lavorare ancora per migliorare il servizio offerto.
Oltre a ciò, bisogna tenere presente che il dataset dal quale BEX recupera le informazioni è in continua evoluzione, viene costantemente arricchito di nuovi dati strutturati sui quali si potrebbero costruire nuove funzionalità e *views* da integrare nell'app; ciò rappresenta il principale motivo per il quale sono state fatte molte delle scelte di progettazione descritte in questa trattazione: rendere la struttura dell'applicazione sufficientemente flessibile e modulare da permettere le modifiche e le estensioni che sicuramente arriveranno nei mesi seguenti alla pubblicazione.

Sviluppi futuri

I possibili sviluppi futuri sono molti: alcuni sono già stati pensati, formalizzati e sono in fase di implementazione, altri emergeranno dai *feedbacks* degli utenti e altri ancora saranno possibili quando il dataset integrerà nuove informazioni sui suoi contenuti. Qui di seguito vengono riportate alcune delle idee in cantiere.

Funzionalità di Bookmarking

L'utente, utilizzando l'applicazione e analizzando i diversi risultati di ricerca, ha la possibilità di tenere traccia degli articoli ritenuti interessanti aggiungendoli ad una lista dei preferiti: avrà quindi la possibilità, quando lo ritiene utile, di poter ritornare alla visualizzazione di tali contenuti.

Ricerca per autore

Una tipologia di ricerca molto utilizzata dagli utenti è sicuramente quella per autore: inserire il nome della autore per avere informazioni sulla sua

attività di ricerca. Questa funzionalità è in parte già stata realizzata, ma non è ancora stata resa disponibile nell'applicazione in quanto deve essere adeguatamente testata per poter assicurare un corretto funzionamento ed escludere comportamenti inaspettati dell'applicazione.

Ricerca per keywords

Nella maggior parte dei tipi di pubblicazioni scientifiche è presente una sezione "keywords" con una lista di parole che si riferiscono agli argomenti principali trattati nell'articolo; queste informazioni si prestano perfettamente ad una nuova funzione di ricerca per parole chiave, ma tali informazioni non sono attualmente disponibili nel *Semantic Lancet Triplestore*.

Nuovi filtri

L'applicazione allo stato attuale mette a disposizione alcune fondamentali tipologie di filtro applicabili alla bibliografia; dai feedback degli utenti ci si aspetta di poter raccogliere utili informazioni su come renderne più intuitivo l'utilizzo o magari ottenere spunti per nuovi filtri avanzati.
Una modifica di facile implementazione è rappresentata dai filtri sui risultati di ricerca (non solo sulle bibliografie): essi non sono stati integrati nell'applicazione in quanto non ritenuti essenziali per la prima versione della stessa; si stanno valutando le modalità con le quali essi potrebbero essere aggiunti nella prossima release di BEX.

Data Visualization

Potrebbe essere molto utile aggiungere nuove *views* che permettano all'utente di analizzare rapidamente particolari domini di dati attraverso elementi di *Data Visualization* realizzati utilizzando apposite librerie Javascript, ad esempio D3.js[4].

[4]http://d3js.org/

Gestione degli utenti

BEX è un'applicazione web la cui *application logic* risiede totalmente sul client. Il server ha la sola funzione di rispondere alle richieste di caricamento dell'applicazione; tutte le successive richieste di informazioni sono verso altri server; per questo motivo l'implementazione del server principale è molto minimale.
Un possibile sviluppo potrebbe consistere nell'integrare alcune funzioni server-side, come ad esempio la gestione degli utenti: utilizzando i *web cookies* si potrebbe tenere traccia delle preferenze di utilizzo impostate dai singoli utenti e personalizzare l'ambiente dell'app in base all'utente che effettua le richieste.

Sezione News

Come già detto, il dataset viene costantemente aggiornato; potrebbe quindi risultare interessante aggiungere una sezione "News" nella homepage dell'applicazione in modo da segnalare agli utenti la disponibilità di nuove utili informazioni o funzionalità.

Efficienza

Un software non è mai perfetto e privo di difetti, soprattutto se si tratta della prima versione. BEX è stata adeguatamente testata, abbastanza da poter escludere la possibilità di malfunzionamenti o comportamenti inaspettati durante la navigazione. Tuttavia, durante le fasi di testing, sono stati individuati aspetti su cui si potrebbe lavorare per migliorare l'efficienza in termini di velocità di elaborazione, caricamento dei dati e occupazione di memoria; queste saranno le prime attività eseguite, prima dell'implementazione di qualsiasi altra funzionalità.

Migliore integrazione con il browser

Si sta attualmente valutando la possibilità di integrare nuove librerie javascript che permettano una più efficiente e profonda integrazione con il bro-

wser, in particolare con i meccanismi di *history*; per effettuare queste operazioni, però, sarà necessario prestare particolare attenzione alla compatibilità (e retrocompatibilità) con i più diffusi browsers.

Note finali

In seguito allo sviluppo di BEX è stato pubblicato un articolo scientifico che ne descrive le funzionalità; per ulteriori informazioni sull'articolo si faccia riferimento a [DGP15].
Il progetto è *open source* e il codice sorgente dell'applicazione è disponibile al seguente indirizzo: https://github.com/lele92/SLP_WebApp

Bibliografia

[ARM00] William Y. Arms, Digital libraries. MIT Press, 2000.

[OS99] Charles Oppenheim, Daniel Smithson. What is the hybrid library?, Journal of information science, 1999

[MAL98] Michael Malinconico, Biblioteche digitali: prospettive e sviluppo, Bollettino AIB, 1998, http://bollettino.aib.it/article/view/8394/7498

[CR02] Fabio Ciotti, Gino Roncaglia, Il mondo digitale: introduzione ai nuovi media, Laterza, 2002

[GAR10] Jesse James Garrett, The Elements of User Experience: User-Centered Design for the Web and Beyond (2nd Edition), Pearson, 2010

[BOR93] Christine Borgman, National electronic library report, in Source-book on digital libraries: report for the national science foundation, Edward A. Fox. Blacksburg: Computer Science Department, 1993

[ABO97] William Y. Arms, Christophe Blanchi, Edward A. Overly, An architecture for information in digital libraries, D-Lib Magazine, 1997

[DGP15] Angelo Di Iorio, Raffaele Giannella, Francesco Poggi, Fabio Vitali, Exploring bibliographies for research-related tasks, 2015

[RSP93] Daniel M Russell, Mark J Stefik, Peter Pirolli, Stuart K Card, The cost structure of sensemaking , Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems, 1993

[WEI96] K. E. Weick, Sensemaking in organizations, Newbury Park, 1996

[SHA07] Nikhil Sharma, Sensemaking: Bringing theories and tools together, 2007

[BHL01] Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, 2001

[AH08] G. Antoniou, F. van Harmelen, A Semantic Web Primer (second edition), 2008

[SIG02] O. Signore, RDF per la rappresentazione della conoscenza, 2002

[CAN05] Daniela Canali, Web semantico e ontologie, 2005, http://www.bibliotecheoggi.it/2005/20050505001.pdf

[GRUB93] T. R. Gruber, A translation approach to portable ontologies. In Knowledge Acquisition, 1993

[BOR97] W. Borst, Construction of Engineering Ontologies, 1997

[W3C14] http://www.w3.org/TR/rdf-schema/ - ultima visita: 25 Febbraio 2015

[PRS94] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey, Human-Computer Interaction, Addison Wesley Publishing Company, 1994

[ABD98] G. Abowd, R. Beale, A. Dix, J. Finlay, Human Computer Interaction, Prentice, 1998

[SHP10] B. Shneiderman, C. Plaisant, Designing the User Interface: Strategies for Effective Human-Computer Interaction: Fifth Edition, Addison-Wesley Publ. Co., 2010

[NM90] J. Nielsen, R. Molich, Heuristic evaluation of user interfaces, Proceeding CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1990

[HHS15] http://www.usability.gov - ultima visita: 25 febbraio 2015

[NIE00] Jakob Nielsen, End of Web Design, 2000, http://www.nngroup.com/articles/end-of-web-design/

[SHN96] B. Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, Proceedings of the IEEE Symposium on Visual Languages, IEEE Computer Society Press, 1996.

[SGR14] Shyam Seshadri, Brad Green, AngularJS: Up and Running.Enhanced Productivity with Structured Web Apps, O'Reilly Media, 2014

[LER13] Ari Lerner, Ng-Book: The Complete Book on Angularjs, 2013

[GOO15] https://docs.angularjs.org/tutorial, Google, 2015 - ultima visita: 25 Febbraio 2015

[THI15] http://thinkster.io/, Thinkster - ultima visita: 25 Febbraio 2015

[W3S15] http://www.w3schools.com/, W3School - ultima visita: 25 Febbraio 2015

[BHB09] C. Bizer, T. Heath, T. Berners-Lee, Linked data - the story so far, 2009

[BER06] Tim Berners Lee, Linked Data, 2009, http://www.w3.org/DesignIssues/LinkedData.html

[SHO09] David Shotton, Semantic Publishing: the coming revolution in scientific journal publishing, 2009

[BCD14] A. Bagnacani, P. Ciancarini, A. Di Iorio, A. G.Nuzzolese, S. Peroni, F.Vitali, The Semantic Lancet Project: a Linked Open Dataset for Scholarly Publishing, 2014

[SHO13] D. Shotton, Publishing: Open citations, 2014

[GMC13] L. García-Castro, C. McLaughlin, Biotea: RDFizing PubMed Central in support for the paper as an interface to the Web of Data, 2013

[IFL09] Functional Requirements for Bibliographic Records Final Report, International Federation of Library Associations and institutions, 1998, http://www.ifla.org/files/assets/cataloguing/frbr/frbr.pdf

[PER14] Silvio Peroni, Semantic Web Technologies and Legal Scholarly Publishing, Springer, 2014

Ringraziamenti

Questo lavoro di tesi conclude un percorso di studi durato tre anni: in questo periodo ho imparato molte cose, non solo sui libri (o davanti allo schermo), ho fatto tante esperienze e tante cose sono cambiate.
Ma non tutto è cambiato, ci sono persone che sono sempre state lì al mio fianco ed è a loro che vanno i miei più grandi ringraziamenti.
Ringrazio mia madre e mio padre che mi sostengono, incoraggiano e mi dimostrano sempre affetto, anche con i piccoli gesti quotidiani; non mi hanno mai fatto mancare nulla e mi sono sempre vicini nei momenti di difficoltà: grazie perchè siete sempre presenti e credete in me.
Mio fratello Luigi è sempre stato pronto a tendermi una mano per aiutarmi, ha una faccia di bronzo ma un cuore d'oro. Forse non gliel'ho mai detto, ma ho grande stima di lui e lo ringrazio per essere il fratello maggiore che chiunque vorrebbe avere: grazie sbirro!
Vorrei ringraziare Gaia: nonostante le centinaia di chilometri che in questi anni ci hanno separato per la maggior parte del tempo, lei mi è stata sempre vicina, giorno per giorno; santa donna, ha saputo tener testa alla mia testardaggine e caparbietà, sono il primo ad ammettere che non è facile! ci siamo sempre aiutati e sostenuti a vicenda e senza di lei questi anni non sarebbero stati così belli.
Un ringraziamento particolare va anche ad Andrea Aquino, il migliore degli amici; con la sua passione per l'informatica e con la sua genialità è stato per me una guida e fonte di ispirazione.
Grazie a tutti gli amici, ai miei coinquilini preferiti, Carlo e Stefano, ai com-

pagni di corso e compagni di progetto: con voi mi sono divertito e avete contribuito a rendere indimenticabili questi anni bolognesi.
Ringrazio il Prof. Fabio Vitali, Dott. Angelo Di Iorio, Dott. Silvio Peroni e Dott. Francesco Poggi per avermi seguito in tutte le fasi del progetto e per avermi dato utili consigli. In particolare vorrei ringraziare Angelo Di Iorio per la pazienza e per il costante sostegno morale.

Raffaele Giannella