Università degli Studi di Padova
Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

La Ricerca a Faccette

Aspetti teorici e pratici

Candidato: Andrea Marcato

Relatore: Ch.mo Prof. Federico Filira

---

28 Settembre 2010
Anno accademico 2009/2010

Indice

Introduzione **1**

I Aspetti teorici 2

1 Cenni storici **2**

1.1 Classificazione: l'albero di Aristotele 2

1.2 La Colon Classification di Ranganathan 3

2 Sistemi di ricerca tradizionali **5**

2.1 Ricerca d'insieme (set retrieval) 5

2.2 Ricerca classificata (ranked retrieval) 5

2.3 Navigazione a directory 6

3 Sistemi di ricerca a faccette **6**

3.1 Ricerca parametrica 7

3.2 Navigazione a faccette 7

3.3 Ricerca a faccette 8

II Aspetti implementativi 11

4 Criteri per la costituzione di un sistema di classificazione a faccette **11**

4.1 Un punto di partenza: i classici 11

4.2 Il modello di analisi semplificato di Spiteri 13

4.3 Creazione del sistema 16

4.4 Esempio di classificazione: detersivi per piatti 17

5 Soluzioni ai problemi di gestione (back-end) **21**

5.1 Dimensioni e storage 21

5.2 Efficienza 22

5.3 Sovraccarico informativo 23

5.4 Disponibilità di metadati 24

6 Soluzioni ai problemi di presentazione (front-end) **25**

6.1 Dove e quando visualizzare le faccette 25

6.2 Organizzazione delle faccette e dei foci 26

6.3 Il campo di ricerca 27
6.4 Selezioni multiple su singola faccetta 28

7 Siti Web che implementano la ricerca a faccette **29**
7.1 Amazon 29
7.2 Yahoo! Shopping 29
7.3 IEEE Xplore 31
7.4 WorldCat 31

III La soluzione open source **35**

8 Apache Solr **35**
8.1 Caratteristiche generali 35
8.2 Confronto con le basi di dati 36
8.3 Configurazione 37
8.4 Indicizzazione e reperimento dei dati 40
8.5 Solr e la ricerca a faccette 41

9 La piattaforma TYPO3 **44**
9.1 Caratteristiche generali 44
9.2 Backend e Frontend 45

10 Integrazione TYPO3 e Solr **46**
10.1 Installazione del software 46
10.2 Configurazione di TYPO3 47
10.3 Considerazioni e sviluppi futuri 50

11 Conclusioni **52**

Introduzione

Ognuno di noi negli ultimi quindici anni è stato testimone dello sviluppo prorompente della scienza e della tecnologia volte al reperimento delle informazioni. Basti pensare che oggi è possibile digitare il nome di una persona o di un'impresa sul motore di ricerca più diffuso al mondo, Google, per essere indirizzati quasi all'istante al suo sito Web o al suo profilo su un qualche *social network*. Per di più, grazie a strumenti quali l'enciclopedia "aperta" Wikipedia, è possibile raggiungere un risultato simile anche riguardo interrogazioni di carattere più ampio, riguardanti i temi più disparati.

Tuttavia i moderni motori di ricerca sono riusciti ad affrontare in maniera adeguata soltanto il problema della ricerca di un oggetto specifico (questione chiamata storicamente dai biblioteconomi *known item search*): in questa situazione l'utente conosce e sa definire ciò che sta cercando [1]. Al contrario, sono ancora in fase di ricerca e sviluppo degli strumenti maturi a supporto della cosiddetta *exploratory search* (trad. *ricerca esplorativa*), che l'utente si trova a compiere o quando non conosce esattamente il dominio della ricerca, o quando non ha chiaro il modo efficace con cui compiere tale ricerca oppure anche quando non sa nemmeno lui stesso cosa sta cercando [2].

La *classificazione a faccette* (in inglese *faceted classification*) si inserisce in questo quadro e gioca un ruolo fondamentale, in quanto supera la debolezza delle prime organizzazioni della conoscenza, ovvero la rigidità degli schemi strettamente gerarchici, introducendo un approccio fedele, da un lato, alla varietà del sapere, ma anche adatto ad un uso concreto. Ciò nonostante, la classificazione a faccette risolve solo il problema della *rappresentazione* dell'informazione, mentre anche le applicazioni più comuni richiedono pur sempre *l'accesso* e *l'uso* di tali informazioni. È a questo punto che entra in gioco la *ricerca a faccette* (in inglese *faceted search*), l'argomento centrale di questo studio.

Parte I

Aspetti teorici

In questa prima parte della tesi verrà presentata la ricerca a faccette nei suoi aspetti storici e teorici. Inizialmente se ne introdurranno i concetti fondamentali a partire da un'esposizione di come in passato sia stata affrontata in generale l'organizzazione della conoscenza; poi si farà una rapida panoramica sui vari approcci al problema del reperimento dell'informazione e infine ci si concentrerà sulle varie tecniche di *information retrival* che affiancano specificamente la classificazione a faccette.

1 Cenni storici

1.1 Classificazione: l'albero di Aristotele

Il più antico sistema di organizzazione della conoscenza fu introdotto da Aristotele. Nelle opere *Historia Animalium* [3], *De Partibus Animalium* [4] e *De Generatione Animalium* [5] il filosofo si occupò della classificazione degli esseri viventi e li divise principalmente in due gruppi: piante e animali; gli animali furono divisi in "dotati di sangue" e "senza sangue"; gli animali con sangue si articolavano ulteriormente in "vivipari" e "ovipari", e via dicendo.

Il lavoro di Aristotele ebbe molto successo e perdurò per circa duemila anni; si può affermare che, in un certo senso, egli fu il primo tassonomista, in quanto organizzò le informazioni in maniera gerarchica. Il termine tassonomia deriva infatti dalle parole greche ταξις (*taxis*), "ordinamento" e νομος (*nomos*) "norma" o "regola" e originariamente era riferito proprio alla classificazione dei viventi, come nel caso della tassonomia Aristotelica o di quella Linneana del *Systema Naturae*, proposta dal XVIII secolo da Carlo Linneo [6]. Tuttavia al giorno d'oggi la parola tassonomia si può riferire in senso più ampio a ogni tipo di classificazione a schema gerarchico e ai principi che lo regolano.

Dunque nell'accezione più moderna, una tassonomia è una qualsiasi organizzazione di entità o concetti in una gerarchia strutturata ad albero. Il nodo radice rappresenta la categoria più generale che descrive l'intero insieme di oggetti; per esempio nella tassonomia Aristotelica il nodo radice corrisponde all'insieme di tutti gli esseri viventi. Di figlio in figlio ogni livello viene diviso in sottoinsiemi fino ad arrivare ai nodi foglia che rappresentano gli oggetti stessi .

La proprietà fondamentale di una tassonomia è insita nella sua struttura stessa: infatti per ogni oggetto o insieme di oggetti che corrispondono ad un nodo, esiste uno e un solo percorso che unisce quel nodo al nodo radice. È per questo motivo che una tassonomia impone un rigido ordinamento logico alle informazioni che rappresenta: un nodo può avere molti figli, ma ogni nodo (esclusa la radice) possiede sempre e solo un genitore.

Tassonomie moderne, analoghe a quelle di Aristotele e Linneo, comprendono, ad esempio, la Classificazione Decimale di Dewey [8] per le biblioteche, ma anche *web directory* come quella di Google [9], Yahoo! [10]o l'Open Directory Project [11] che mirano a catalogare un ampio numero di siti internet.

L'ordinamento imposto da una tassonomia può essere però poco flessibile. In particolare, il requisito che ogni cammino dal nodo radice a un nodo foglia debba essere unico è un vincolo molto rigido, il che obbliga analisti e progettisti a compiere scelte tanto difficili quanto più composito è l'oggetto o l'argomento da catalogare. Ad esempio "Storia d'Europa" è figlio di "Storia" o di "Europa"?

La classificazione a faccette fornisce una soluzione elegante a questi problemi.

1.2 La Colon Classification di Ranganathan

La Classificazione Decimale di Dewey (DDC) è ancora usata tutt'oggi da molte biblioteche (ad esempio quella del dipartimento di Ingegneria dell'Informazione). Un nodo nella tassonomia della DDC è composto da una sequenza di cifre decimali in cui ogni cifra successiva rappresenta una diramazione dell'albero. Per esempio, il cammino al nodo "Gatti" è così strutturato [12]:

600 Tecnologia (Scienze applicate)
630 Agricoltura e tecnologie affini
636 Allevamento
636.8 Gatti

Risulta strano, secondo questa classificazione, che i gatti siano sotto "Tecnologia". In realtà i gatti risiedono sia in 638.8 sotto "Allevamento", sia in 599.75 sotto "Zoologia" (in quanto 500 è "Scienza"). Rimane all'arbitrio dell'analista decidere come vengano catalogati i libri riguardo, ad esempio, il *comportamento* dei gatti.

Questo problema fu di grande interesse per Shiyali Ramamrita Ranganathan, un matematico e bibliotecario indiano, che mise in discussione l' "ospitalità" [1] della notazione DDC. Egli affermo che l'ospitalità del DDC risiede solamente all'estremità destra

[1]ovvero la possibilità di aggiungere una nuova diramazione in una tassonomia

della notazione, cioè, l'unico modo per aggiungere una nuova categoria è quello di assegnare un nodo figlio ad uno già esistente. Ciò si traduce nel specificare la cifra meno significativa relativa ad un nodo: o cambiando uno zero (0) in un'altra cifra, o aggiungendo una nuova cifra a destra del punto decimale.

Ranganathan superò questa limitazione introducendo nel 1933 [13] un sistema di classificazione che rispecchiava la decomposizione di un concetto in *faccette*. Egli divise la conoscenza in cinque macro categorie (cioè, le faccette, chiamate originariamente *isolates*) - personalità, materia, energia, spazio e tempo (PMEST). Queste categorie sono mutualmente esclusive, cioé non si sovrappongono (si dicono quindi *ortogonali*); ogni faccetta poi è strutturata gerarchicamente per divisione e le varie gerarchie possono essere combinate tra loro per formare argomenti compositi multidimensionali. La notazione introdotta da Ranganathan rappresenta un argomento composito come una serie di faccette (espresse come sequenza di lettere e numeri) separate dai due punti (e altra punteggiatura a mano a mano che lo schema continuò ad espandersi), da qui il nome *Colon Classification*.

Esempio (proposto originariamente da Ranganathan stesso [14]):
L2153:4725:63129:B28 rappresenta uno "studio statistico sul trattamento del cancro al palato molle tramite radio".

Questo argomento composito può essere suddiviso nelle sue quattro faccette che lo compongono, ognuna delle quali è organizzata gerarchicamente:

- Medicina (L) → Apparato digerente (L2) → Bocca (L21) → Palato (L215) → Palato molle (L2153)
- Malattia (4) → Malattia strutturale (47) → Tumore (472) → Cancro (4725)
- Cura (6) → Cura attraverso sostanze chimiche (63) → Cura attraverso un elemento chimico del 2° gruppo (6312) → Cura con il radio (63129)
- Studio matematico (B) → Studio algebrico (B2) → Studio statistico (B28)

Si può vedere che questo sistema è molto più flessibile del DDC o di qualsiasi altro schema basato su una gerarchia unica.

2 Sistemi di ricerca tradizionali

Dopo aver esposto le caratteristiche dei sistemi di rappresentazione delle informazioni, in questa sezione si presenterà brevemente il problema di come rendere tali informazioni accessibili.

2.1 Ricerca d'insieme (set retrieval)

I primi motori di ricerca funzionavano in maniera molto diversa da quelli moderni. Essi infatti impiegavano un modello detto di "ricerca d'insieme" (*set retrieval*), il quale a ogni interrogazione (*query*) restituiva un insieme di documenti disordinato, in quanto non esisteva ancora il concetto di ordinamento dei risultati in base alla pertinenza.

Questo modello è detto anche di "ricerca booleana", in quanto prevede che l'utente formuli interrogazioni utilizzando operatori booleani (AND, OR, NOT) [15]. Molti sistemi del genere prevedono l'uso di una sintassi booleana estesa in modo da includere operatori che specificano l'ordine in cui appaiono i termini cercati nel testo o la loro vicinanza ad altre parole all'interno di un documento. Alcuni sistemi permettono anche di specificare dove un termine cercato si deve trovare (ad esempio nel campo del titolo oppure in quello dell'autore) [16].

Questo sistema di ricerca è molto flessibile, ma soffre di un difetto fondamentale: è poco intuitivo, e ciò può spiegare il motivo per cui i motori di ricerca moderni adottano altri sistemi.

2.2 Ricerca classificata (ranked retrieval)

L'alternativa alla complessità della ricerca booleana si può trovare nei motori di ricerca odierni, che implementano la cosiddetta "ricerca classificata" (o meglio: *ranked retrieval*).

Nel studiare un'alternativa al set retrieval i ricercatori scelsero una via totalmente diversa: essi sollevarono l'utente dall'onere di immettere query complesse, strutturate e molto formali dal punto di vista sintattico e scelsero un approccio basato su interrogazioni non strutturate (o *free-text*, a testo libero). Invece di restiuire un insieme di risultati che soddisfano la richiesta con precisione il ranked retrieval si basa sull'ordinamento in base alla pertinenza, per cui i risultati primi in classifica sono più attinenti alla query rispetto agli ultimi.

Senza dubbio, query di questo tipo risultano essere più facili da formulare dagli utenti rispetto alle espressioni booleane, ma ciò comporta lo svantaggio che le inter-

rogazioni non rappresentino più un filtro ben definito da applicare all'insieme dei documenti, bensì esse diventano un metro di paragone con cui ogni documento viene confrontato e ordinato secondo il grado di pertinenza.

Alcuni esempri di ranked retrieval applicato a insiemi di documenti ipertestuali sono l'algoritmo HITS di Jon Kleinberg e l'algoritmo PageRank di Larry Page e Sergey Brin - quest'ultimo servì come fondamenta per il motore di ricerca Google.

2.3 Navigazione a directory

Negli ultimi anni la ricerca tramite interrogazioni a testo libero è diventata ormai un'abitudine. Ciò nonostante, nel campo del reperimento dell'informazione questa interfaccia non è l'unica esistente.

In precedenza si è discusso delle tassonomie come un approccio alla rappresentazione delle informazioni che ha anticipato di due millenni l'avvento di Internet. Una tassonomia però può rendersi anche utile dal momento che la sua organizzazione intrinseca rende le informazioni contenute accessibili e ricercabili. Esempi d'uso di una tassonomia per consentire l'accesso alle informazioni sono i directory di Yahoo e dell'Open Directory Project .

I directory offrono un grande vantaggio rispetto ai metodi di ricerca tramite query testuale visti in precedenza, indipendentemente dal tipo di implementazione. Infatti, poiché il contenuto di un directory è organizzato gerarchicamente, l'utente viene guidato verso un sottoinsieme di documenti potenzialmente attinenti all'informazione che cerca. L'utente, di volta in volta, elabora e raffina i suoi bisogni, imparando a correggere il tiro grazie alle scelte disponibili.

D'altro canto questo modello soffre delle limitazioni delle tassonomie esposte poc'anzi: l'utente deve scoprire il percorso per giungere ad una certa informazione nello stesso modo in cui è stata concipita da colui che l'ha classificata. Ciò che può essere intuitivo per un progettista può non esserlo per gli utenti o persino per altri progettisti.

3 Sistemi di ricerca a faccette

Dopo aver presentato i problemi relativi alla classificazione delle informazioni e le soluzioni tradizionali per l'accesso ad esse, ci si concentrerà ora sulle tecniche di information retrieval incentrate sulla classificazione a faccette (o *multidimensionale*) dei documenti. Prima di discutere della ricerca a faccette, si consideranno i suoi predecessori: la ricerca parametrica a la navigazione a faccette.

3.1 Ricerca parametrica

La ricerca parametrica si concretizza sostanzialmente in un'interfaccia grafica per la ricera booleana applicata a un'insieme di documenti classificati a faccette. L'interfaccia permette agli utenti di formulare interrogazioni specificando in maniera visuale le varie condizioni sui valori delle faccette. La query risultante sarà composta da AND e OR: valori multipli, selezionati all'interno di una singola faccetta, vengono collegati mediante OR logico, mentre i vincoli su faccette diverse vengono combinati usando l'AND. Il sistema infine risponde con un insieme di elementi che soddisfano le condizioni imposte nella query.

Il modo in cui le opzioni sono presentate dipende dalla natura delle faccette. Per faccette composte da valori enumerabili (esempio: una lista di produttori di vino) ha senso mostrare la lista delle opzioni, eventualmente espandibile per evitare di sommergere l'utente di informazioni.

Per faccette organizzate in maniera gerarchica (esempio: una locazione geografica articolata in continente, stato e regione) potrebbe rivelarsi vantaggioso per l'utente poter navigare l'albero dalla radice verso il basso e poter selezionare non solo nodi foglia, ma anche nodi interni.

Per faccette numeriche (esempio: prezzi) l'utente molto probabilmente vorrebbe poter selezionare un intervallo di valori, magari con un'indicazione sui limiti massimi e minimi ammissibili. A seconda della sofisticatezza dell'interfaccia varia il livello libertà con con cui l'utente può effettuare le sue scelte.

3.2 Navigazione a faccette

Come si è visto, la ricerca parametrica implementa una forma di set retrieval: infatti essa offre un sottoinsieme delle funzionalità della ricerca booleana, applicata alle faccette piuttosto che a del testo non strutturato. Come la ricerca booleana però, la ricerca parametrica presenta lo stesso problema riguardo la modalità di formulazione delle query: se il numero di vincoli imposti dall'utente risulta eccessivo, la ricerca potrebbe avere esito negativo, al contrario se l'utente non è abbastanza specifico il sistema potrebbe restituire anche troppi risultati.

La navigazione a faccette introduce ciò che manca alla ricerca parametrica: una guida. La ricerca parametrica infatti richiede che l'utente definisca tutti vincoli in una sola volta, selezionando i valori d'interesse su tutte le faccette disponibili; al contrario, la navigazione a faccette permette all'utente di sviluppare l'interrogazione progressi-

vamente, vedendo di volta in volta l'effetto che la scelta di un valore su una faccetta produce sui valori disponibili per le altre faccette.

Dal punto di vista dell'utente, la navigazione a faccette elimina la possibilità di sovraspecificare la query e non ottenere alcun risultato. Ma come fare nel caso di dati testuali?

3.3 Ricerca a faccette

I sistemi di ricerca tradizionali sono stati progettati per trattare documenti puramente testuali, mentre, al contrario, le tecniche di ricerca parametrica e navigazione a faccette descritte in precedenza ignorano il testo non strutturato e assumono che i documenti siano costituiti da un insieme di valori classificati a faccette.

Tipicamente, le situazioni reali richiedono di trattare insiemi di documenti semi-strutturati, ovvero contenenti una parte di testo non strutturato e una parte di attributi strutturati, detti anche anche *metadati*. In greco μετά (meta) significa "oltre" o "dopo" ma usato come prefisso può significare anche "riguardo a" [17]. Dunque i metadati di un documento non sono altro che degli attributi strutturati che riguardano il documento stesso e sono logicamente memorizzati insieme ad esso [18].

Se la parte strutturata di un documento è stata progettata per essere conforme a un sistema di classificazione a faccette, allora è possibile combinare la ricerca testuale, applicata alla parte non strutturata del documento, con la navigazione a faccette sui metadati. Questo è il concetto chiave della ricerca a faccette.

Un esempio di ricerca a faccette si può vedere in Figura 3.1 e mostra il sito di commercio di vini K&L. Con questo nuovo approccio l'utente non inizia da subito a usare le faccette, ma piuttosto prima esegue una ricerca a testo libero sulle descrizioni dei vini, e poi utilizza i metadati per raffinare la query selezionando il paese di provenienza, l'intervallo di prezzi e così via. Il sistema di ricerca impiegato assegna le seguenti faccette ad ogni vino in vendita: Varietà (es: Cabernet, Pinot Nero, Chardonnay, ecc...), Nazione, Sotto-Regione, Prezzo, Annata, Tipo (es: rosso, bianco, frizzante ecc...), Descrizione Speciale (es: in arrivo, in magazzino, rarità, più venduto, ecc...), Dimensione Bottiglia, Valutazione (voto da 1 a 100) e Locazione (del magazzino). L'interfaccia presenta le faccette numeriche Prezzo e Valutazione in raggruppamenti, mentre quelle più numerose possono essere espanse tramite il link "View More"; in ogni momento della ricerca, l'interfaccia permette di eliminare le precedenti selezioni sulle faccette senza seguire un ordine predeterminato.

Più avanti si analizzeranno in dettaglio altri siti che implementano la ricerca a faccette, mentre nella prossima parte dello studio ci si concentrerà sugli aspetti pratici da considerare in fase di progettazione di un sistema reale.

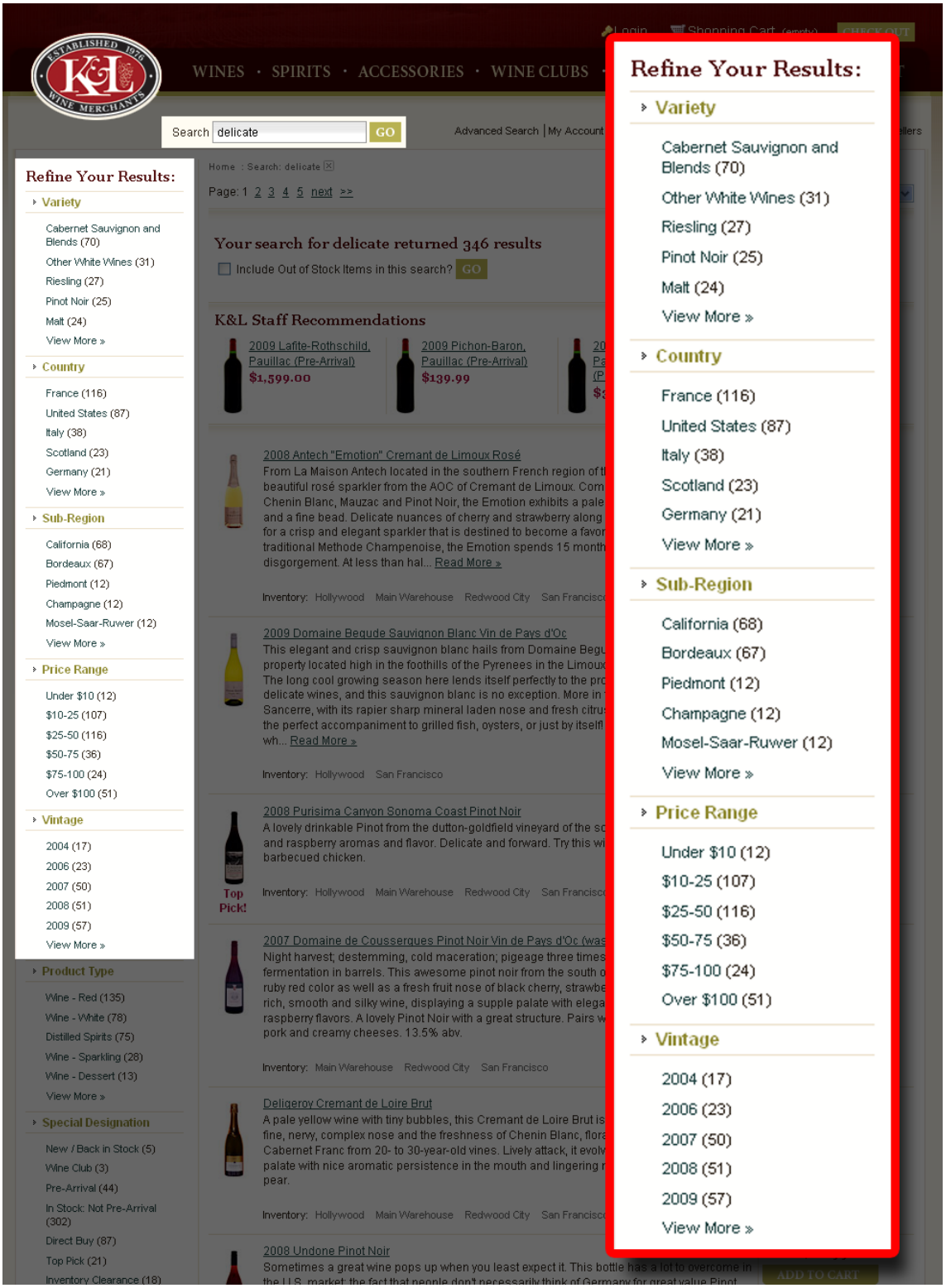

Figura 3.1: Pagina di ricerca del sito K&L Wines

Parte II

Aspetti implementativi

Dopo aver esposto in linea generale il problema della classificazione ci si occuperà ora, nel concreto, di come sia possibile progettare un sistema di classificazione a faccette. Partendo da quanto originariamente proposto da Ranganathan per l'analisi dei concetti, si passerà al metodo operativo più semplice di Spiteri e si farà un esempio tratto dalla vita quotidiana. Infine si affronteranno gli aspetti legati alla realizzazione di un sistema di ricerca informatico dei dati catalogati (come un sito internet), sia dal punto di vista della gestione che della presentazione.

4 Criteri per la costituzione di un sistema di classificazione a faccette

4.1 Un punto di partenza: i classici

Per iniziare a muovere i primi passi nella progettazione di un sistema di classificazione multidimensionale è utile prima volgere lo sguardo ai sistemi esistenti, per osservare la loro struttura e le idee che ne stanno alla base. Si comincierà quindi dai più conosciuti sistemi "universali" quali la Colon Classification di Ramganathan, già citato in precedenza, e la seconda edizione della Bliss Bibliographic Classification (BC2) [19].

Come già accennato, la Colon Classification utilizza cinque faccette per catalogare le entità di interesse:

- Personalità: l'oggetto centrale in questione (es. un evento o una persona in una classificazione storica, oppure un animale in una classificazione zoologica).
- Materia: i componenti e le proprietà dell'oggetto.
- Energia: le caratteristiche dinamiche dei processi che lo interessano.
- Spazio: elementi geografici o spaziali relativi ad esso.
- Tempo: le sue fasi cronologiche.

Le classiche faccette proposte da Ranganathan vennero riprese e ampliate nel 1977 dalla BC2 che fornisce tredici categorie per l'analisi e l'organizzazione degli oggetti. Esse vengono spiegate con chiarezza da Vanda Broughton [20]:

- Oggetto [inglese *thing*]: equivalente alla categoria Personalità di Ranganathan, riguarda l'interesse principale o l'oggetto di qualsiasi disciplina (le piante in botanica, le sostanze in chimica, le nazioni in storia). Questa categoria contiene di norma oggetti fisici o loro aggregazioni.

- Tipo: questa categoria, indicante in generale una relazione genere-specie con Oggetto, contiene raggruppamenti generali ampi di concetti (es. "strumenti a fiato", come Tipo dell'oggetto "strumenti musicali"). Nelle gerarchie tassonomiche in cui le relazioni genere-specie sono in gran parte permanenti e predefinite (botanica, zoologia), questa categoria può essere ridondante.

- Parte: componenti e sottosistemi di Oggetto (es. l'Oggetto "bicicletta" ha per Parti: ruote, pedali, freni, gomme; l'Oggetto "cellula" ha per Parti: nuclei, vacuoli, apparati di Golgi).

- Proprietà: proprietà e caratteristiche dell'Oggetto. Talvolta difficili da distinguere da Tipo in alcune circostanze, ma generalmente hanno natura astratta piuttosto che concreta (es. l'Oggetto "bicicletta" ha per Tipi: da montagna, da corsa, da turismo, mentre per Proprietà: peso, efficienza, velocità).

- Materiale: equivalente alla categoria Materia della Colon Classification. Rappresentato da materiali grezzi, componenti ed elementi, è più fondamentale di Parte (es. l'Oggetto "casa" ha per Parti: tetto, muri, finestre, fondamenta, i quali consistono dei Materiali: legno, mattoni, tegole, vetro).

- Processo: prima fra le due categorie di energia o attività, Processo è rappresentato da azioni intrinseche e spontanee all'interno di entità o sistemi; azioni che "avvengono da sole". Esempi sono solitamente i verbi intransitivi (o gli equivalenti nominali), come: crescita, cambiamento, malattia, flusso.

- Operazione: azioni determinate da un agente esterno; azioni che vengono "fatte a" un'entità o sistema dall'esterno. Esempi sono solitamente i verbi transitivi, come: sperimentare, tagliare, costruire, mangiare.

- Prodotto: esiti o risultati di processi in, o di operazioni su, entità; solitamente consistono in prodotti fisici, come cibi, farmaci o tessuti in agricoltura e orticoltura. Questa categoria è in gran parte limitata all'area della tecnologia, ed è generalmente assente nelle arti e nelle discipline umanistiche e sociali.

- Sottoprodotto: autoesplicativo; analogamente alla precedente, è un'altra categoria tecnologica.

- Paziente: il destinatario di operazioni, quando è diverso dalla categoria Oggetto o entità. I casi sono in gran parte tecnologici; per esempio: alcuni buchi (Paziente) possono essere impressi (Operazione) in componenti (Parte) per macchinari.

- Agente: i mezzi attraverso i quali delle Operazioni vengono effettuate; gli Agenti possono essere in genere distinti in persone e strumenti o attrezzature, e a un livello complesso possono essere rappresentati da istituzioni. Le due categorie di agenti possono presentarsi insieme; es. in medicina un chirurgo (Agente persona) può asportare del tessuto utilizzando un laser (Agente strumento).

- Spazio: qualsiasi tipo di dimensione politica, fisiografica o spaziale (es.: gli USA, montuoso, interno).

- Tempo: qualsiasi tipo di caratteristica storica, cronologica o temporale (es.: medievale, permanente, notturno).

Risulta evidente come l'utilizzo di queste categorie permetta di analizzare concetti di pressoché tutte le aree della conoscenza (ed è per questo che la Colon Classification e la BC2 sono dette classficazioni *universali*). Tuttavia la stessa Broughton affermò che, concentrandosi su domini più ristretti, sicuramente esistano molte altre faccette, come ad esempio "genere" e "forma" nell'ambito letterario. Dunque l'approccio di Ranganathan e il successivo raffinamento del BC2 forniscono un eccellente punto di partenza per la progettazione di un sistema di questo tipo, ma le loro categorie possono essere comunque modificate per adattarsi a qualsiasi bisogno particolare e il livello di dettaglio impiegato può essere tanto elevato quanto più limitato è il dominio di interesse per la specifica applicazione.

4.2 Il modello di analisi semplificato di Spiteri

Ranganathan presentò una serie dettagliata di 46 canoni, 13 postulati e 22 principi da seguire per analizzare i concetti e costituire una classificazione multidimensionale [21]. Queste regole complesse e articolate sono state analizzate da Louise Spiteri [22], che le ripropose in una forma più concisa, comprensibile ed insegnabile.

I principi enunciati da Spiteri sono semplici ma al tempo stesso esaustivi. Il processo di classificazione viene diviso in tre parti:

- il **Livello Ideale** (*Idea Plane*), impiegato nell'analisi delle entità e delle loro componenti;

- il **Livello Verbale** (*Verbal Plane*), che si occupa di scegliere la giusta terminologia per esprimere le componenti delle entità;

- il **Livello Notazionale** (*Notational Plane*) che esprime le suddette componenti tramite una notazione appropriata.

Data la natura delle moderne tecnologie e del World Wide Web, il Livello Notazionale risulta meno importante degli altri in questa sede.

I principi di Spiteri per la creazione di classificazioni a faccette sono i seguenti:

Livello Ideale: Principi per la scelta delle faccette

a) **Differenzazione**: quando si divide un'entità in parti componenti, bisogna ricorrere a caratteristiche (cioè le faccette) che rendano i componenti risultanti nettamente distinti. Esempio: l'entità "esseri umani" può essere divisa secondo la caratteristica "sesso" poichè ciò produrrà due componenti distinte: "maschi" e "femmine".

b) **Rilevanza**: è necessario che le faccette scelte riflettano lo scopo del sistema di classificazione.

c) **Accertabilità**: bisogna scegliere faccette che siano accertabili e ben definite.

d) **Permanenza**: le faccette dovrebbero rappresentare qualità permanenti dell'oggetto in analisi.

e) **Omogeneità**: le faccette dovrebbero poter essere applicabili a tutti gli oggetti del dominio in analisi.

f) **Mutua Esclusività**: tutte le faccette dovrebbero rappresentare una sola caratteristica di divisione dell'oggetto e non dovrebbero sovrapporsi.

g) **Categorie Fondamentali**: non esistono delle categorie fondamentali per tutti gli ambiti; le categorie dovrebbero essere derivate in base alla natura di ciò che si sta classificando.

Livello Ideale: Principi per l'ordinamento delle Faccette e dei Foci

a) **Sequenza Rilevante**: l'ordine di presentazione delle faccette e la strutturazione dei termini che le compongono (chiamati nella letteratura *foci*, plurale di *focus*), dovrebbero essere in accordo con la natura, l'argomento e il dominio del sistema di classificazione. Esempi: ordine cronologico, alfabetico, spaziale o geometrico, da semplice a complesso, da complesso a semplice, secondo quantità crescente o decrescente.

b) **Sequenza Consistente**: stabilito un determinato ordine delle faccette, esso non dovrebbe essere modificato se non cambia anche lo scopo, l'argomento e il dominio del sistema. [Questo principio può essere ignorato nel caso il sistema preveda un riordinamento delle faccette.]

Livello Verbale

a) **Contesto**: il significato di un termine individuale è dato dal contesto, che dipende dalla posizione del termine all'interno del sistema di classificazione. Esempio: le città Londra, in Inghilterra, e Londra, in Ontario, Stati Uniti, possono essere entrambe identificate solo dal termine "Londra" e a quale delle due esso si riferisca dipende se si trova nella parte del sistema che classifica le città dell'Inghilterra oppure quelle dell'Ontario.

b) **Attualità**: la terminologia adottata in un sistema di classificazzione dovrebbe riflettere quella utilizzata correntemente nell'ambito di interesse. Ciò significa che è necessario prestare costante attenzione al sistema, che può richiedere revisioni periodiche.

Livello Notazionale

a) **Sinonimi**: ogni argomento deve essere rappresentato da un solo unico numero identificatore di classe.

b) **Omonimi**: ogni numero identificatore di classe può rappresentare un solo argomento.

c) **Ospitalità**: la notazione deve permettere l'aggiunta di nuovi argomenti, faccette e termini in ogni punto del sistema di classificazione.

d) **Ordinamento**: la notazione deve riflettere l'ordinamento degli argomenti che sta alla base del sistema di classificazione.

Rispettando queste regole è possibile creare un sistema di classificazione multidimensionale, coerente e flessibile.

4.3 Creazione del sistema

I principi di Spiteri guidano il progettista nell'accurata scelta delle faccette da usare per la classificazione, ma i passi effettivi da compiere per la realizzazione dell'intero sistema sono stati razionalizzati da Vickery [23] e completati da Denton [24]. Quest'ultimo prevede la seguente serie di operazioni:

1. **Campionamento**. Si raccoglie un campione rappresentativo di entità nel dominio di interesse: nel caso di dominio ampio è necessario utilizzare un numero sufficientemente elevato di campioni in modo tale da coprire ogni possibilità; nel caso di dominio limitato, si usano tutte le entità a disposizione.

2. **Elenco delle Entità**. Si redige un elenco delle entità; si divide la descrizione di ognuna in parti individuando e separando tra loro i concetti fondamentali.

3. **Scelta delle Faccette**. Si esaminano i concetti fin qui trovati e si trovano delle categorie generali "di alto livello" che accumunano le varie entità. Queste categorie devono essere poi ulteriormente analizzate e decomposte per raffinamenti successivi fino a trovare un insieme di faccette, mutuamente esclusive ed esaustive nel complesso, con le quali sia possibile descrivere tutte le entità. Questa fase si colloca nel Piano Ideale, quindi è necessario seguire le linee guida dei Principi per la Scelta delle Faccette di Spiteri, facendo sempre riferimento allo scopo del sistema di classificazione e ai suoi futuri utilizzatori.

4. **Strutturazione delle Faccette**.

    - Si effettua un ordinamento preliminare dei foci per ogni faccetta, usando come linee guida i Principi per l'Ordinamento delle Faccette e dei Foci, e si controlla che tutte le entità campione possano essere classificate usando quei termini. In caso contrario si ritorna all'analisi del punto precedente. Questa fase si colloca sia nel Piano Ideale che in quello Verbale, dunque si devono seguire i relativi principi.

- Ogni faccetta può essere poi sviluppata o espansa secondo una propria logica e una propria struttura. Denton cita a riguardo l'esempio della Kwasnick [25]: in una classificazione di Arte e Architettura la faccetta "Periodo" può essere organizzata in ordine cronologico, "Materiale" in ordine gerarchico, mentre "Luogo" può essere stutturata ad albero.
- Infine si adotta un vocabolario standardizzato (*controlled vocabulary*): i foci infatti devono rappresentare concetti in maniera biunivoca, quindi tutte le parole o termini d'uso comune che hanno lo stesso significato devono essere condensati in un unico focus. La risultante mutua corrispondenza fra termine e concetto, fra significante e significato, dovrebbe garantire un sistema coerente privo di ridondanze o ambiguità. Anche per questa operazione risulta fondamentale avere sempre chiaro il target del sistema e scegliere come foci quei termini che rendano intuitiva la navigazione dell'utente. Un buon principio di progettazione da attuare è: *Don't make users think*.

5. **Ordinamento delle Faccette**. Questa operazione riguarda il Livello Notazionale. Si sceglie una faccetta che identificherà la caratteristica o attributo principale delle entità e poi si darà un ordine di visualizzazione ragionevole alle altre. Oggi, le tecnologie alla base del web dinamico permettono agli utenti esperti di riordinare le faccette a piacimento, apportando modifiche personalizzate alle interfacce, ma probabilmente la maggior parte delle persone utilizzerà le viste standard.

6. **Classificazione**. A questo punto il sistema di classificazione è completato. Si possono quindi classificare tutte le entità nel dominio di interesse.

7. **Revisione, test e manutenzione**. Nel caso di problemi nel punto precedente, dovuti a incompletezze o imprecisioni, è necessario tornare indietro alla fase di analisi. Se possibile, può rivelarsi utile effettuare un test del sistema su un campione di utenti finali. Infine è molto importante aggiornare la terminologia, se essa va incontro a cambiamenti, e controllare che nel tempo gli insiemi di faccette e foci impiegati siano sempre sufficientemente completi rispetto al dominio di interesse da classificare.

4.4 Esempio di classificazione: detersivi per piatti

Per chiarificare gli aspetti teorici, si applicheranno i passi del processo descritto poc'anzi ad un esempio concreto.

Il dominio di interesse è formato da un insieme ristretto di prodotti commerciali per il lavaggio delle stoviglie, ritrovabili in un comune supermercato. Si suppone che il target sia composto dai consumatori, dunque la categorizzazione dovrà essere coerente alle loro esigenze, e non, ad esempio, a quelle di ingegneri o chimici, che potrebbero avere bisogno di una classificazione più tecnica: il sistema servirà a guidare il consumatore nella scelta del detersivo più adatto secondo le proprie necessità. Per semplicità, si è omessa la catalogazione di prezzo e quantità per ogni detersivo, dati che invece potrebbero essere indispensabili per un'implementazione concreta.

4.4.1 Campionamento

La fase di campionamento si è svolta in un supermercato di una catena che offre una gamma abbastanza variegata di prodotti. Essendo il dominio comunque limitato, l'analisi avverrà su tutti gli elementi che lo compongono.

4.4.2 Elenco delle Entità

La fase di elenco delle entità quindi ha riguardato i seguenti prodotti. La suddivisione riportata rispecchia quella trovata negli scaffali del supermercato stesso.

- Detersivi per lavaggio a mano liquidi: Scala Piatti; Splendid Concentrato, Effetto Balsamo; Svelto Gel Piatti, Gel Attivo; Despar Piatti; Nelsen Concentrato Cristalli di Sale, Argilla Naturale, Acidi di Frutta, Carboni Attivi, Limone; Sole Piatti; Dixan Piatti con Calcio, con Aceto, con Aloe Vera.

- Detersivi per lavastoviglie liquidi: Finish Power Gel Tutto IN1 Gel, Power Gel con Tecnologia Elimina Odori, Power Gel Classic; Pril Perfect Gel.

- Detersivi per lavastoviglie in polvere: Despar Lavastoviglie in Polvere al Limone, Lavastoviglie in Polvere Tripla Forza; Finish Polvere Limone.

- Detersivi per lavastoviglie in pastiglia: Finish Powerball Quantum, Powerball Quantum Limone, Powerball Classic, Powerball Tutto In Uno Limone, Powerball Tutto In Uno Arancia; Despar Detergente Lavastoviglie, Detergente Lavastoviglie Tutto In 1; Pril 5, 10 Ultra-Brillante; Fairy Tutto In 1, Tutto In 1 Platinum.

4.4.3 Scelta delle Faccette

La scelta delle faccette prevede un'iniziale suddivisione "di alto livello". Gli aspetti che vengono subito in mente dai nomi stessi dei prodotti sono: Marca (Despar, Fi-

nish, Svelto ecc...), Tipo (a mano liquido, lavastoviglie liquido, lavastoviglie polvere, lavastoviglie pastiglie) e Profumo (arancia, limone, melograno, ecc...).

Una più approfondita analisi rivela che, in realtà, Tipo contiene in se' due informazioni: quella sulla forma (liquido, polvere, pastiglie) e quella su ciò che compie l'azione vera e propria di lavaggio. Rifacendosi alla BC2, quest'ultima può essere identificata dalla faccetta Agente (mano, lavastoviglie).

Marca, Forma, Agente e Profumo sono un buon punto d'inizio per la classificazione, ma non dicono nulla sulle diverse proprietà di lavaggio che caratterizzano i vari prodotti. Scorrendo le descrizioni delle confezioni, i detersivi possono citare le seguenti caratteristiche aggiuntive: azione brillantante, azione detergente, azione disincrostante, azione sgrassante, anticalcare, antiodore, delicato sulle mani, effetto ammollo, effetto balsamo, effetto pretrattaggio, efficace a basse temperature, efficace su lavaggi brevi, elimina odori, funzione del sale, igienizzante, protezione dei bicchieri, protezione dell'argento, protezione del vetro, salvavetro, sciogligrasso.

Bisogna notare una sottile differenza concettuale tra queste proprietà: alcune si riferiscono a effetti del detersivo sulle stoviglie (come "protezione dei bicchieri") altri a effetti sull'Agente (come "delicato sulle mani" o "anticalcare"), dunque tali foci possono essere assegnati a due faccette: Proprietà Speciale ed Effetto sull'Agente. I nomi qui scelti sono opinabili, ma esprimono bene i concetti.

4.4.4 Strutturazione delle Faccette

Le faccette trovate a questo punto sono: Marca, Forma, Agente, Effetto sull'Agente, Profumo e Proprietà Speciale. Alcuni dei foci di Effetto sull'Agente e Proprietà Speciale sono tra loro equivalenti (come "sciogligrasso" e "azione sgrassante" , "azione del sale" e "anticalcare", "effetto balsamo" e "delicato sulle mani", "effetto ammollo" e "effetto pretrattaggio") e quindi vanno condensati in foci univoci dal nome quanto più comprensibile. Inoltre il focus "azione detergente" si può eliminare perché, com'è ovvio, tutti i detersivi hanno "azione detergente".

Infine, l'unico ordine ragionevole da dare ai foci è quello alfabetico, nessun altro aiuterebbe di più gli utenti; quindi le faccette risultanti e loro valori sono:

- Marca: Despar, Dixan, Fairy, Finish, Nelsen, Pril, S-Budget, Scala, Sole, Splendid, Svelto.
- Forma: liquido, pastiglia, polvere.
- Agente: lavastoviglie, mano.

- Effetto sull'Agente: anticalcare, antiodore, delicato sulle mani.

- Profumo: aloe, arancia, bergamotto, fiori di limone, fiori di loto, limone, limone verde, melograno.

- Proprietà Speciale: azione brillantante, azione disincrostante, azione sgrassante, effetto ammollo, efficace a basse temperature, efficace su lavaggi brevi, igienizzante, protezione dell'argento, protezione del vetro.

Ordinamento delle Faccette

Va effettuata una riflessione sull'ordine in cui vengono presentate le faccette: "Marca, Forma, Agente, Effetto sull'Agente, Profumo, Proprietà Speciale" è un ordine che può soddisfare l'utenza target del sistema? Seguendo il Principio di Sequenza Rilevante, si può affermare che la prima cosa a cui una persona pensa nel comprare un detersivo è se esso sarà destinato al lavaggio manuale o in lavastoviglie, quindi Agente può essere messo al primo posto. Poi l'attenzione andrà sulla Forma e probabilmente alla Marca, in quanto ognuno ha sempre una qualche preferenza, e magari, dopo, Profumo può essere di un certo interesse; infine possono seguire Effetto sull'Agente e Proprietà Speciale. Questo sarà l'ordine predefinito, ma un'implementazione concreta potrebbe permettere agli utenti di riordinare le faccette a piacimento.

Quindi lo schema definitivo di classificazione è:

- Agente: lavastoviglie, mano.

- Forma: liquido, pastiglia, polvere.

- Marca: Despar, Dixan, Fairy, Finish, Nelsen, Pril, S-Budget, Scala, Sole, Splendid, Svelto.

- Profumo: aloe, arancia, bergamotto, fiori di limone, fiori di loto, limone, limone verde, melograno.

- Effetto sull'Agente: anticalcare, antiodore, delicato sulle mani.

- Proprietà Speciale: azione brillantante, azione disincrostante, azione sgrassante, effetto ammollo, efficace a basse temperature, efficace su lavaggi brevi, igienizzante, protezione dell'argento, protezione del vetro.

4.4.5 Classificazione

Ora è possibile classificare le entità nel dominio di interesse. Ad esempio:

Dixan Piatti con Aceto	
Agente	mano
Forma	liquido
Marca	Dixan
Profumo	melograno, fiori di limone
Effetto sull'Agente	(nessuno)
Proprietà Speciale	azione brillantante

Finish PowerGel con Tecnologia Eliminaodori	
Agente	lavastoviglie
Forma	liquido
Marca	Finish
Profumo	limone
Effetto sull'Agente	anticalcare, antiodore
Proprietà Speciale	azione brillantante, efficace sui lavaggi brevi

La fase di Revisione, Test e Manutenzione, per i fini di questo esempio, non è necessaria in quanto non ci sono utenti da consultare.

5 Soluzioni ai problemi di gestione (back-end)

5.1 Dimensioni e storage

Come visto nei capitoli precedenti, la ricerca all'interno di un sistema di classificazione multidimensionale rende la collezione di documenti esplorabile dall'utente. Dunque un sistema di ricerca a faccette offre potenzialmente maggiore utilità, se applicato ad un

insieme vasto di documenti, rispetto ad un sistema di ricerca tradizionale, poiché in pratica permette l'accesso ad un maggior numero di informazioni.

Purtroppo maggiori sono le dimensioni in gioco, maggiori sono i costi da sostenere, sia in termini di requisiti hardware sia in termini di complessità computazionale. Le dimensioni di un sistema di classificazione multimensionale riguardano:

- Il numero di entità / documenti.
- Il numero di faccette e foci di ogni documento.
- Il testo ricercabile di ogni documento.

Sebbene i precisi requisiti di storage dipendano dalle strutture dati effettivamente impiegate, in prima approssimazione la dimensione del sistema sarà data dal prodotto tra il numero di documenti classificati e la quantità media di spazio occupato dai valori delle faccette e del testo ricercabile di ognuno.

Una semplice implementazione potrebbe utilizzare due tabelle: un indice inverso che associa ciascun valore assunto dalle faccette alla lista dei documenti che contengono quel termine e una tabella di documenti che associa ogni documento ai valori delle faccette.

5.2 Efficienza

La ricerca a faccette è onerosa dal punto di vista computazionale. Infatti, quando un sistema si appresta ad elaborare una query, come primo passo determina l'insieme di documenti che soddisfa i parametri dell'interrogazione, operazione semplice che può essere portata a termine con varie tecniche di set retrieval sull'indice inverso. Poi però, il passo successivo consiste nel visualizzare i foci disponibili per raffinare l'insieme dei risultati. Inoltre, molte applicazioni mostrano, accanto ad ogni focus, anche il numero di risultati del potenziale raffinamento. È intuibile quindi come le operazioni riguardo i possibili raffinamenti siano più complesse rispetto al solo calcolo dei risultati.

I possibili raffinamenti possono essere calcolati in due modi: secondo un approccio top-down oppure bottom-up.

- L'approccio top-down utilizza l'indice inverso: per ogni focus viene calcolata l'intersezione tra l'insieme dei documenti che lo contengono e l'insieme dei risultati della ricerca.
- L'approccio bottom-up invece parte dalle entità presenti nell'insieme dei risultati e per ognuna prevede l'iterazione sulle faccette per contare la frequenza dei foci.

Entrambi gli approcci sono onerosi, e la loro complessità dipende dalle strutture dati usate per implementare l'indice inverso e i documenti.

L'efficienza di un sistema informatico per la ricerca a faccette riguarda le seguenti metriche:

- Throughput, ovvero numero medio di query elaborate al secondo. Questo aspetto può essere facilmente migliorato distribuendo il carico di lavoro su più server, ognuno dei quali dovrà lavorare su una copia dell'intero insieme di documenti. Questa soluzione introduce però problemi di integrità dei dati.

- Latenza, ovvero tempo percepito dall'utente tra l'invio della query e la ricezione del risultato. Anche in questo caso, la distribuzione del carico di lavoro su più macchine può migliorare le prestazioni, grazie all'*intra-query paralellism*, una tecnica che scompone la query in parti che possono essere elaborate separatamente; la necessità di sincronizzazione più macchine distinte può essere aggirata impiegando un server con diversi processori multi-core in grado di elaborare più thread contemporaneamente, anche se ciò può portare a situazioni di contesa delle risorse; per di più, la latenza introdotta necessariamente dal mezzo fisico della rete potrebbe comunque prevalere sui vantaggi introdotti da questa soluzione.

Infine, un fattore influente sull'efficienza di un sistema è dato dalla frequenza degli aggiornamenti dei dati e dalla conseguente latenza di aggiornamento, ovvero il tempo trascorso tra l'inserimento dei nuovi dati, l'indicizzazione e la loro effettiva presenza nei risultati.

5.3 Sovraccarico informativo

La grande quantità di informazioni potenzialmente accessibili mediante un sistema di ricerca a faccette può rischiare di sommergere gli utenti in fase di raffinamento delle scelte. Dunque, è necessaria una riflessione non solo su *come* presentare le informazioni (questione relativa al front-end, affrontata nella Sezione 6.2), ma anche su *cosa* mostrare all'utente.

I due fattori da considerare in questo ambito sono: il numero di faccette e il numero di foci. Teoricamente essistono infiniti modi per classificare un'insieme di entità; nella pratica invece il numero di faccette è finito, anche se a volte può essere molto grande; inoltre esiste il problema delle dipendenze tra le faccette: il principio di Omogeneità di Spiteri può essere non sempre soddisfatto nei casi in cui alcune faccette si applicano solo ad un sottoinsieme delle entità classificate. Nella realtà infatti si lavora con un

gran numero di faccette che in alcuni casi sono eterogenee e interdipendenti, quindi di volta in volta è necessario effettuare una scelta su cosa mostrare e cosa nascondere.

Alcuni accorgimenti, dunque, per non sovraccaricare l'utente di informazioni sono i seguenti:

- Favorire la visualizzazione delle faccette i cui valori sono assegnati a tutti i documenti nell'insieme dei risultati.
- Favorire la visualizzazione delle faccette i cui valori sono uniformemente distribuiti nell'insieme dei risultati.
- Consolidare, in fase di progettazione, le faccette che contengono valori simili e che possono sovrapporsi.

Per quanto riguarda il problema in caso di numero elevato di foci, la soluzione ideale sarebbe l'organizzazione dei valori in gerarchie ad albero, in modo tale da mostrare inizialmente il nodo radice, corrispondente alla categorizzazione più generale dei valori, e scendere progressivamente ad ogni raffinamento. Questa soluzione però smette di funzionare nel caso in cui l'albero ottenuto sia di un'altezza tale da portare a problemi di usabilità di sistema.

Nei casi, poi, in cui non sia nemmeno possibile ottenere una ragionevole strutturazione gerarchica dei foci, è necessario ricorrere ad accorgimenti quali:

- Visualizzazione dei foci più ricorrenti nell'insieme dei risultati.
- Visualizzazione dei foci più ricorrenti nell'insieme dei risultati che in tutta la collezione di documenti.
- Suddivisione dei valori in base a prefissi (es. A-H, I-R, S-Z) o limiti numerici.
- Raggruppamento dei valori grazie a statistiche di similarità.

5.4 Disponibilità di metadati

La ricerca a faccette si basa sull'esistenza di un qualche insieme di documenti organizzato secondo un sistema di classificazione multidimensionale. La disponibilità di metadati strutturati a faccette è un requisito irrinunciabile, ma spesso questa condizione viene meno nei casi in cui la costituzione di un sistema di ricerca si basi su un insieme di documenti composti unicamente da testo non strutturato. È universalmente

riconosciuto, dall'esperienza quotidiana, che l'80-85% delle informazioni di un certo interesse provengano da fonti costituite da testo non strutturato [26].

Esistono comunque varie tecniche, dette di *text mining*, per arricchire il testo non strutturato ed ottenere metadati a faccette; vari approcci usati comprendono l'utilizzo di metadati nascosti ma facilmente reperibili (es. fonte del documento, tipo, lunghezza ecc...) oppure l'analisi statistica basata su regole predeterminate per classificare i documenti in diverse categorie [27], ma la trattazione estensiva di queste tecniche va oltre lo scopo di questo lavoro.

6 Soluzioni ai problemi di presentazione (front-end)

6.1 Dove e quando visualizzare le faccette

Nell'implementazione di un sistema di ricerca multidimensionale, è compito del progettista decidere in che modo debbano essere visualizzati i suoi due elementi portanti, ovvero i risultati e le faccette. Un'applicazione potrebbe mostrarli insieme in un'unica vista, oppure in due viste separate: non esiste una regola universale e le scelte devono essere dettate dallo scopo della specifica applicazione e dal tipo di utenti che dovrà servire.

Per le interfacce utente che presentano sia i risultati che le faccette in un'unica vista, gli approcci convenzionali prevedono il posizionamento delle faccette in un pannello laterale, a sinistra dei risultati, oppure in un pannello direttamente sopra i risultati. Il primo caso spinge gli utenti a focalizzarsi inizialmente sul risultato dell'interrogazione, che occupa molto spazio rispetto alle faccette e attira maggiormente l'attenzione; questo approccio si adatta molto bene alle esigenze di chi vuole arrivare ai risultati in breve tempo ed effettua raramente dei raffinamenti alla query. Il secondo caso invece rende le faccette più visibili e importanti, in quanto esse vengono collocate tra il campo di ricerca e l'insieme dei risultati: lo svantaggio di richiedere uno sforzo in più per arrivare ai risultati viene compensato dal fatto che l'utente, forzato di fronte al pannello di raffinamento, elabora ulteriormente la query anziché volgere lo sguardo subito ai risultati e ciò può determinare un uso più produttivo del sistema di classificazione.

Un'altra soluzione prevede la collocazione dei risultati e delle faccette in due viste diverse. In questo modo il progettista decide quale di esse debba essere mostrata per prima, scelta che comporta in ogni caso dei rischi: infatti, se vengono visualizzati prima i risultati si rischia che l'utente ignori totalmente l'esistenza delle faccette, invece

se si mostrano prima le faccette gli utenti più "pigri" potrebbero essere scoraggiati e abbandonare il sistema.

6.2 Organizzazione delle faccette e dei foci

Il problema di sovraccarico informativo affrontanto nella Sezione 5.3 riguarda anche il front-end del sistema. In questa sede la questione centrale riguarda *come* organizzare al meglio la presentazione delle faccette e dei loro valori.

Le strategie per organizzare le faccette sono simili a quelle esposte in precedenza per il back-end:

- Impiego di un ordine fisso che non cambia durante la navigazione.
- Ordinamento dinamico delle faccette secondo un indice stimato di utilità all'utente.
- Organizzazione in gruppi delle faccette simili.

La scelta tra ordine statico o dinamico richiede una valutazione dei benefici di entrambe le possibilità: da un lato, un ordine statico tende a essere capito e imparato facilmente; l'utente si crea uno schema mentale stabile del sistema, in quanto vede sempre le stesse faccette nello stesso ordine; questa soluzione è applicabile quando il numero delle faccette è limitato ed esse sono sempre visibili.

D'altro canto, l'ordinamento dinamico risulta più efficace di quello statico nei casi in cui il numero di faccette è tale da non essere possibile mostrarle tutte in una sola volta, oppure quando alcune faccette si applicano solo a sottoinsiemi delle entità catalogate.

Infine, il raggruppamento delle faccette simili è reso un'alternativa appetibile grazie a tecnologie quali AJAX (Asynchronous JavaScript and XML) [28] o Adobe Flash [29], con le quali è possibile creare *Rich Internet Applications*. Siti Internet che ne fanno uso presentano interfacce utente interattive che permettono di mostrare o nascondere menu e sottomenu con prontezza, senza dover ricaricare interamente le pagine web.

Strategie simili si applicano anche per la presentazione dei foci:

- Ordine statico indipendente dalla query
- Ordinamento in base all'utilità.
- Visualizzazione progressiva dei foci organizzati gerarchicamente (cioè, un livello alla volta).

L'ultima soluzione può essere messa in pratica anche quando la faccetta non è esplicitamente gerarchica: infatti ci sono casi in cui è possibile introdurre una gerarchia artificiale, in modo da ottenere un albero in cui ogni nodo possiede un numero di figli semplice da gestire.

Ad esempio, è possibile gerarchizzare una faccetta i cui valori sono stringhe di nomi di persona dividendo i cognomi in intervalli: il primo livello sarà formato da A-E, F-K, L-P, Q-U, e V-Z; il nodo A-E può essere diviso in A, B, C, D e E; il nodo A poi può essere diviso in Aa-Ae, Af-Ak ecc... e così via finché ogni nodo non possiede un numero gestibile di figli.

6.3 Il campo di ricerca

Il campo di ricerca sta alla base della potenza della ricerca a faccette, oggetto di questo studio; esso è l'elemento dell'interfaccia che permette all'utente di inserire query semistrutturate, a testo libero, per poi navigare tra i risultati in maniera strutturata, grazie ai raffinamenti tramite faccette.

Il design di un'interfaccia pone comunque delle questioni riguardo al comportamento di questo elemento. Le domande più frequenti di un progettista sono le seguenti:

- *Una nuova ricerca testuale dovrebbe applicarsi ai risultati della ricerca precedente?*
  L'approccio convenzionale è quello di cancellare i risultati della ricerca precedente quando l'utente immette una nuova query, in quanto si presume che avvenga prima la ricerca testuale e poi eventualmente il raffinamento tramite faccette. Nonostante ciò, può essere utile aggiungere accanto al campo di ricerca un opzione che permetta di cercare ulteriormente all'interno dei risultati che si stanno visualizzando.

- *La ricerca dovrebbe avvenire su tutti i campi dei documenti o solo su alcuni?*
  Solitamente i motori di ricerca tradizionali effettuano la ricerca all'interno del testo non strutturato dei documenti, ma ciò può essere un limite in un sistema di classificazione a faccette. L'utente, infatti, potrebbe voler cercare non solo tra il testo dei documenti, ma anche tra i metadati. Il vantaggio di un sistema che tenga conto di questo è che spesso un insieme di documenti che hanno in comune il valore di una certa faccetta è molto più rilevante di un insieme di documenti che contengono il valore di quella faccetta all'interno del loro testo non strutturato.

- *Come dovrebbero essere interpretate di default le query formate da più parole?*
  La domanda può essere riformulata in questo modo: la stringa di testo immessa deve essere intepretata come un insieme di parole combinate dall'operatore OR (cioé, trova qualsiasi parola), dall'operatore AND (cioé, trova tutte le parole), oppure come una frase singola (cioè, trova le parole che compaiono in quella esatta sequenza) e l'utente deve poter scegliere tra queste possibilità? Prassi comune, a cui ormai gli utenti sono abituati, è quella di interpetare query di più parole come una "congiunzione disordinata", per cui i risultati contengono sì tutte le parole, ma necessariamente non nello stesso ordine in cui sono state digitate.

- *Si può effettuare la ricerca anche su varianti delle parole immesse nel campo?*
  Questa tecnica prende il nome di *query expansion*. Spesso il suo impiego minimo permette di inserire sostantivi singolari o plurali indistintamente; un utilizzo più intenso invece ricorre ad un vocabolario di sinonimi per ottenere risultati relativi anche a parole semanticamente simili a quelle immesse dall'utente. Questa tecnica comunque dovrebbe essere impiegata con moderazione in modo da non confondere l'utente con risultati inaspettati o inspiegabili.

- *Il sistema dovrebbe presentare più campi di ricerca, un campo parametrizzato o un'interfaccia di ricerca avanzata?*
  Anche se non esiste una risposta unica, è certo che più campi di ricerca confondono l'utente, pochi utilizzeranno le funzionalità di ricerca avanzata e la maggior parte lascerà le impostazioni predefinite. L'attenzione quindi dovrebbe incentrarsi su un campo di ricerca il cui funzionamento sia il più ragionevole possibile di default, lasciando pannelli di ricerca avanzata a viste separate.

6.4 Selezioni multiple su singola faccetta

La situazione tipica per la ricerca o navigazione a faccette prevede di selezionare al più un singolo valore alla volta per faccetta, ma ci sono casi in cui l'utente potrebbe volerne selezionare più d'uno: ad esempio, la selezione di un intervallo di prezzi o date, che sottointende l'operazione disgiuntiva OR tra i valori, oppure la selezione di più qualità o caratteristiche che sottointende l'operazione congiuntiva AND.

Gli utenti, com'è noto, non sono bravi a dedurre le operazioni booleane che lavorano dietro le quinte delle interfacce, dunque la sfida è quella di riuscire a comunicare in maniera visiva se la selezione di più valori in una faccetta implichi un'operazione

disgiuntiva o congiuntiva. Ciò viene spesso realizzato tramite elementi standard dei moduli delle pagine HTML come check-box o liste a selezione multipla.

Solitamente, le faccette di entità a cui viene assegnato un singolo valore (come "marca" o "potenza elettrica") lavorano bene con operazioni disgiuntive invece, le faccette di entità a cui vengono assegnati più valori (come "Optional di serie" per le automobili o "Periferiche hardware" per i microprocessori) lavorano meglio con operazioni congiuntive

7 Siti Web che implementano la ricerca a faccette

7.1 Amazon

Amazon[2] è una tra le prime compagnie di commercio online, fondata negli Stati Uniti. Dal 1995, anno del lancio, il portale amazon.com si è arricchito di una gamma sempre più ampia di merce: la vendita iniziale di libri è stata affiancata via via da quella di DVD, CD musicali, software, videogiochi, prodotti elettronici, abbigliamento, mobili, cibo, giocattoli e altro, per un numero complessivo di quasi 30 milioni di prodotti acquistabili. Solo il dominio .com è stato visitato nell'ultimo anno da una media di 67 milioni di visitatori unici al mese, con un picco di 79 milioni nel mese di dicembre 2009 [30].

Il vasto numero di prodotti è accessibile grazie ad un sistema di classificazione multidimensionale: le entità sono suddivise in macrocategorie di alto livello, chiamate Dipartimenti (es. "Libri", "MP3", "Casa, Giardino & Animali", "Vestiti & Accessori" ecc...), ognuna delle quali poi ospita un sistema di classificazione a se stante. La Figura 7.1 mostra le faccette mostrate da Amazon a seguito della ricerca della parola chiave "printers".

7.2 Yahoo! Shopping

Yahoo! Shopping[3] è un sito che dal 2004 permette di confrontare tra diversi venditori i prezzi dei prodotti che ospita.

Per la sua natura, la faccetta Prezzo compare al primo posto della lista di quelle disponibili. I prodotti catalogati sono divisi in Dipartimenti; in caso di ricerche con termini ambigui, cioé presenti in più dipartimenti, il sistema prima chiede all'utente

[2]http://www.amazon.com
[3]http://shopping.yahoo.com

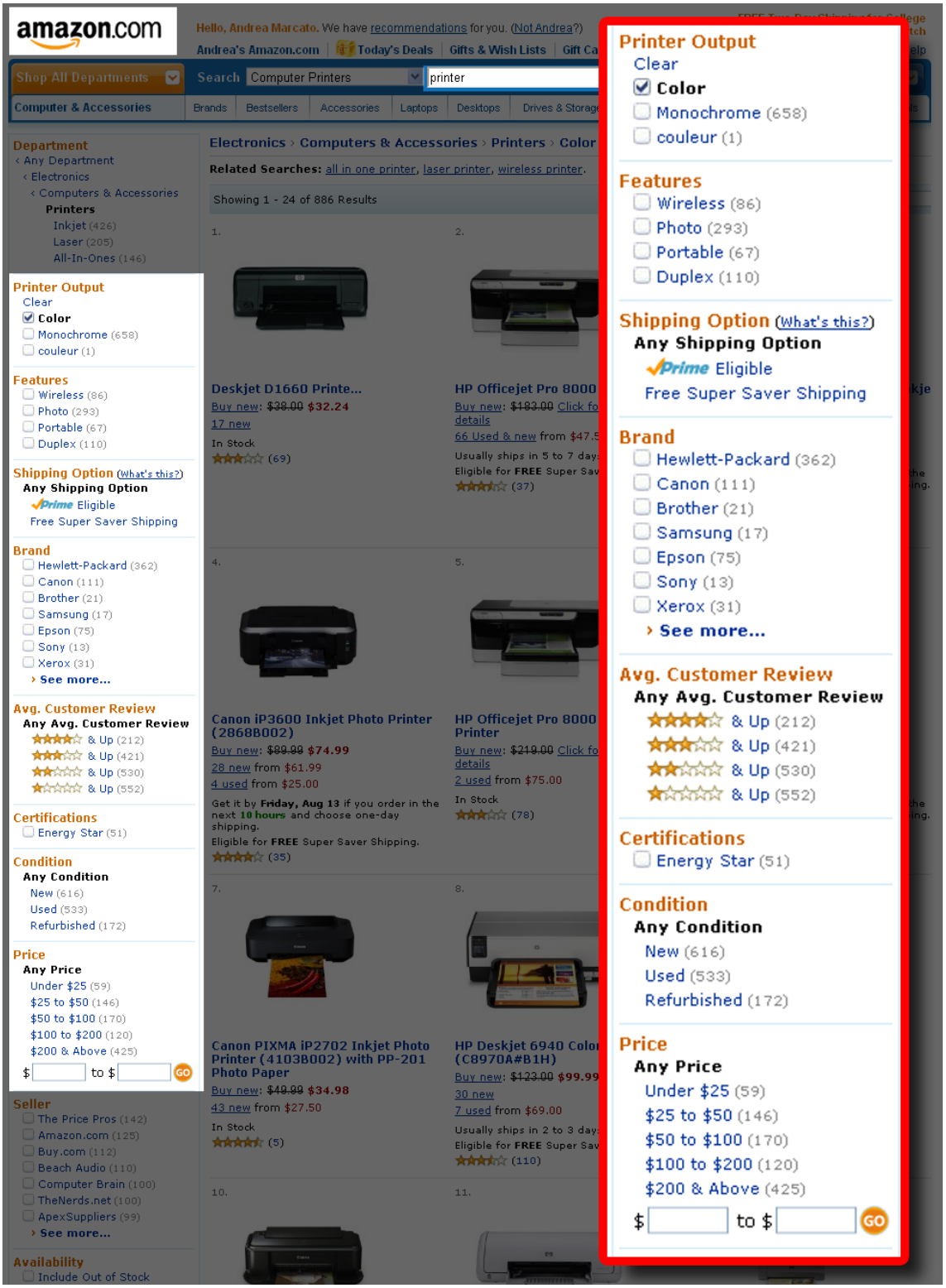

Figura 7.1: Faccette proposte da Amazon.com per la parola chiave “printers”

di scegliere in quale approfondire la ricerca, poi presenta le faccette e i risulati relativi al dipartimento selezionato. La Figura 7.2 mostra le faccette mostrate da Yahoo! Shopping a seguito della ricerca della parola chiave “printers”.

7.3 IEEE Xplore

IEEE Xplore è il sito che permette l’accesso alle pubblicazioni scientifico-tecnologiche prodotte dall’Istituto degli Ingegneri Elettrici ed Elettronici (IEEE) e dai suoi partner. La banca dati del sito ospita più di 2 milioni di articoli, che ben si prestano ad essere catalogati in un sistema multidimensionale.

Come mostrato in figura 7.3, le faccette impiegate sono, in ordine di presentazione: Tipo (conferenze, riviste, libri, corsi ecc...), Anno di Pubblicazione, Autore, Affiliazione, Titolo della Pubblicazione, Editore, Argomento e Luogo della Conferenza.

L’interfaccia prevede il raffinamento della ricerca anche tramite ulteriore testo libero, attraverso il campo “Search within results”; l’anno di pubblicazione può essere specificato come valore singolo o come intervallo attraverso due indicatori posti su una barra di scorrimento orizzontale, oppure inserendo i limiti temporali nei campi “From:” e “To:”; infine, il problema di un possibile alto numero di foci di ogni faccetta viene mitigato dalla possibilità di ricercare con testo libero il valore desiderato all’interno dei foci stessi (senza dover ricaricare la pagina).

7.4 WorldCat

WorldCat[4] è la più grande rete mondiale di contenuti e servizi bibliotecari. Il sito è un motore di ricerca di libri, cd ed altro materiale reperibile nelle biblioteche di tutto il mondo che partecipano alla cooperativa globale *Online Computer Library Center* (OCLC). Al momento il progetto comprende 72.000 biblioteche in 171 paesi; la base di dati ospita quasi 200 milioni di voci bibliografiche e tiene traccia di un totale di 1,6 miliardi di oggetti fisici.

Le faccette impiegate nella classificazione sono: Autore, Formato (es: libro, giornale, mappa, ecc...), Anno, Contenuto (Tesi/Dissertazione, Narrativa, Non-Narrativa, Biografica), Lingua e Argomento (quest’ultima organizzata gerarchicamente). Vedi Figura 7.4.

[4]http://www.worldcat.org

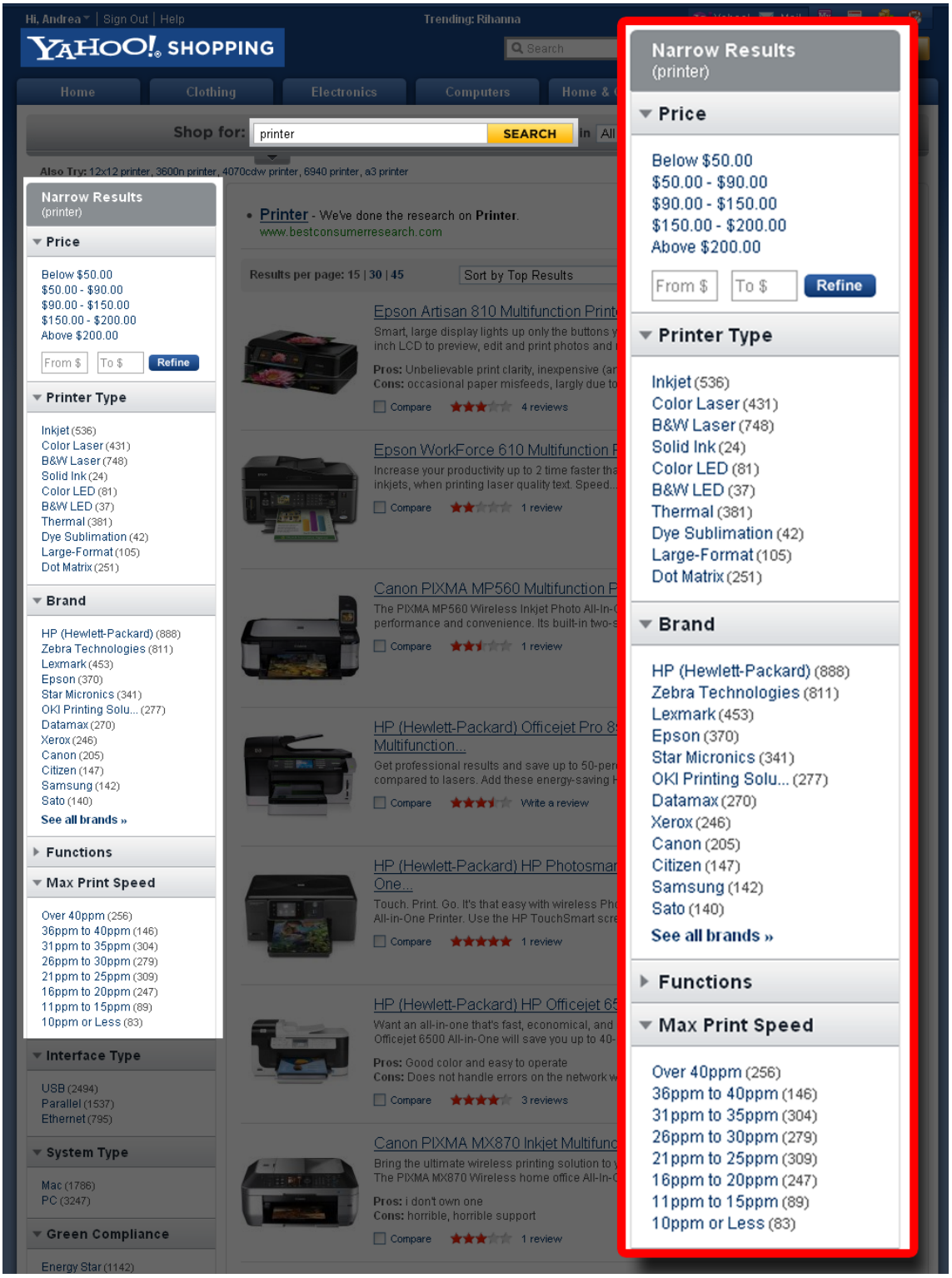

Figura 7.2: Faccette proposte da Yahoo! Shopping per la parola chiave “printers”

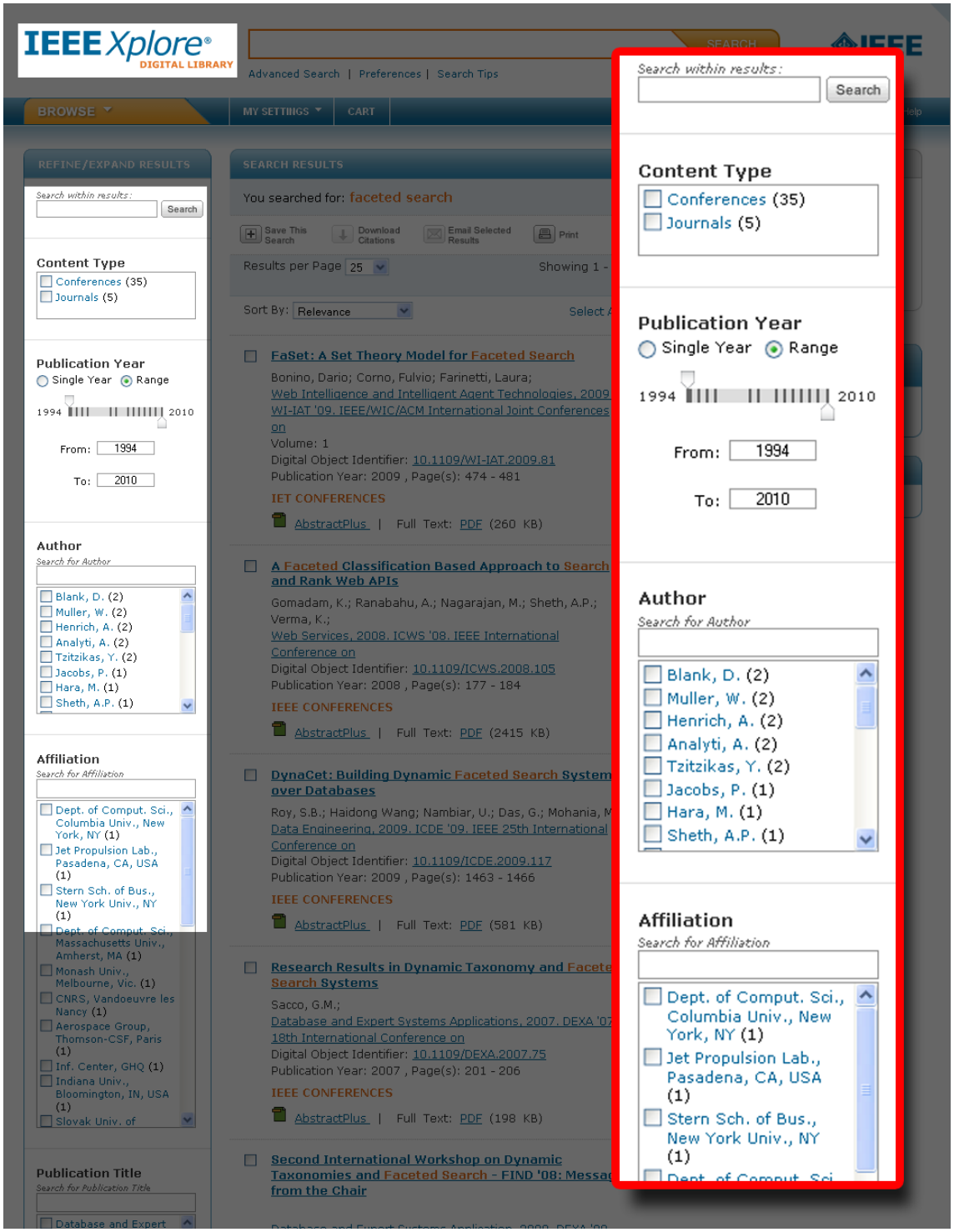

Figura 7.3: Faccette proposte da IEEE Xplore per la parola chiave “faceted search”

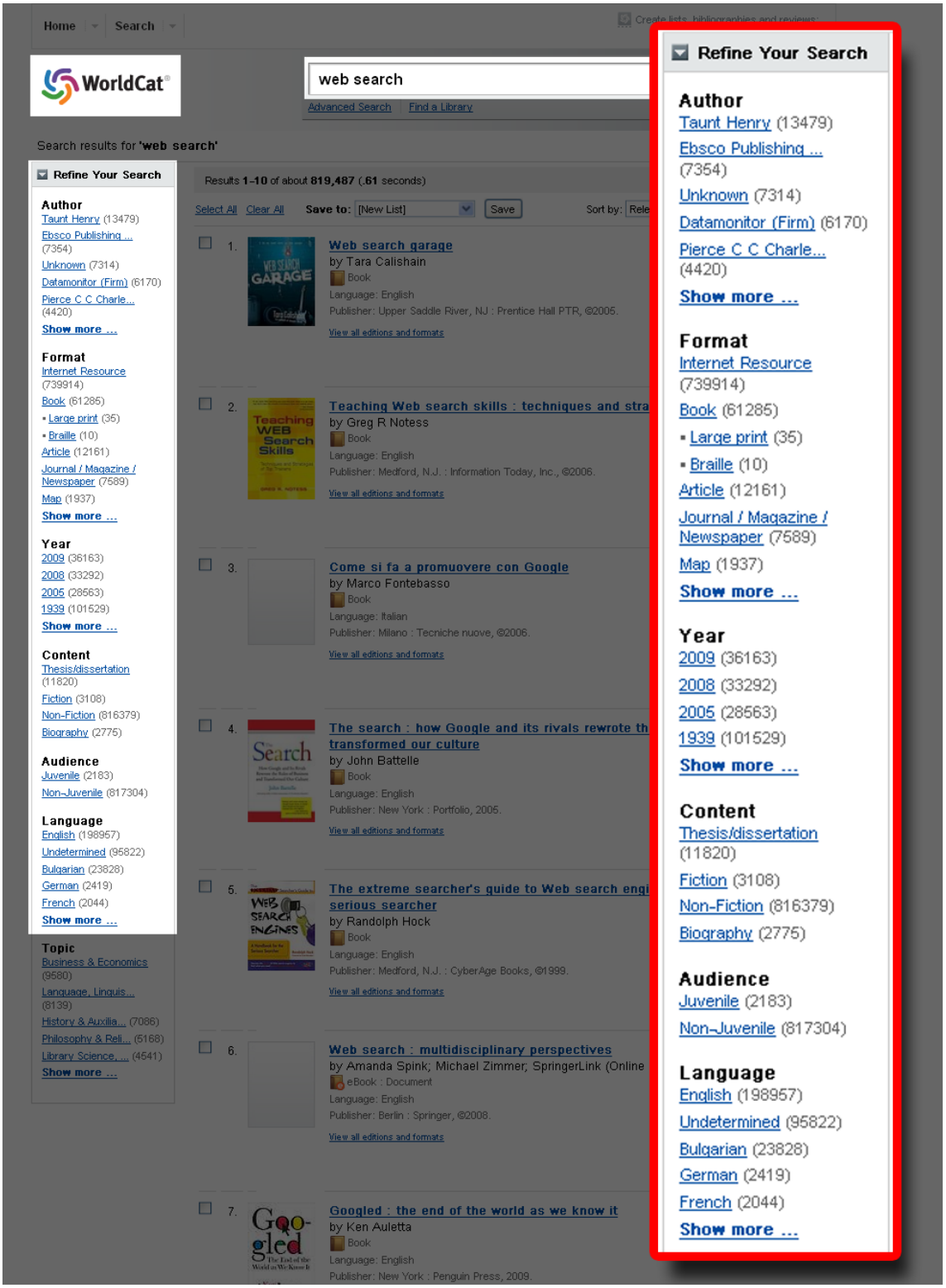

Figura 7.4: Faccette impiegate in WorldCat

Parte III

La soluzione open source

Questa parte dello studio è dedicata all'analisi dei componenti principali che possono essere impiegati in un sistema di ricerca a faccette destinato al web e implementato mediante software open source. Oggetti di questa trattazione saranno il server di ricerca Apache Solr, la piattaforma di *content management* TYPO3 e il software che si occupa della loro integrazione. Questo approfondimento non ha avuto solamente un fine accademico, ma è stato portato a termine con lo scopo di applicare la conoscenza acquisita al progetto di un sito web reale, sviluppato presso Telerete Nordest S.r.l. di Padova.

8 Apache Solr

Apache Solr è la soluzione open source che permette di implementare in concreto la ricerca a faccette. Le informazioni presentate in questa sezione sono state tratte principalmente dal libro-guida di Smiley e Pugh, *Solr 1.4 Enterprie Search Server* [31], a cui si rimanda per una trattazione esaustiva degli argomenti più avanzati.

8.1 Caratteristiche generali

Solr è il motore di ricerca che lavora dietro alle quinte di siti come CNet, SourceForge e Netflix; scritto in Java, usa il protocollo HTTP e XML per la comunicazione e possiede molte funzionalità come l'evidenziazione dei risultati, la correzione ortografica delle query, i suggerimenti automatici su query e documenti simili tra loro e, infine, l'organizzazione a faccette dei risultati.

Solr in realtà è basato su un altro progetto del gruppo Apache, chiamato Lucene, una libreria di codice per la gestione di un motore di ricerca testuale ad alte prestazioni. Alcune tra le più importanti caratteristiche di Lucene sono:

- La memorizzazione di un indice inverso per il rapido reperimento di documenti tramite parole chiave.
- Strumenti per l'analisi del testo che trasformano una stringa di caratteri in una serie di termini (le parole), che sono le unità fondamentali per l'indicizzazione e la ricerca.

- Una ricca sintassi del linguaggio di interrogazione.
- Un algoritmo di *scoring* flessibile e personalizzabile, per l'ordinamento dei risultati in base alla rilevanza.
- La possibilità di evidenziare le parole trovate.
- Un correttore ortografico per query basato sui dati indicizzati.

Lucene di per se' non è un server, né esso è in grado di comunicare tramite XML. Dunque Solr può essere visto come l'implementazione sotto forma di server di questa libreria, anche se in realtà vengono introdotte delle caratteristiche aggiuntive, come le seguenti.

- Indicizzazione e interrogazione tramite protocollo HTTP e XML: la comunicazione con Solr avviene inviando i dati all'URL della servlet attraverso il metodo POST del protocollo HTTP.
- Diverse memorie di cache per ottenere i risultati più rapidamente.
- Un'interfaccia di amministrazione accessibile via browser che include statistiche sull'uso della cache, un modulo di interrogazione dell'indice, un browser dello schema dati e il dettaglio degli algoritmi di scoring e di analisi del testo.
- File di configurazione per lo schema e per il server stesso (in XML).
- Visualizzazione delle faccette e dei foci dei documenti trovati con relativo contatore.
- Possibilità di scalare il sistema distribuendolo su più macchine.

Solr viene distribuito sotto forma di servlet e dunque deve essere installato ed eseguito su un server per applicazioni web che implementino le specifiche Java Servlet di Oracle, come Apache Tomcat [32] o Jetty [33].

8.2 Confronto con le basi di dati

La principale peculiarità di Lucene consiste nell'adottare un unico indice che può essere visto come una base di dati con una singola tabella, sulla quale però le operazioni relazionali (come i JOIN) non sono possibili. Questo indice viene usato solo per la ricerca, e di solito non dovrebbe essere la fonte primaria di dati per le applicazioni;

esso presenta parecchie ridondanze, e, cosa molto importante, può ospitare campi multi valore.

Altre differenze rispetto alle basi di dati relazionali riguardano:

- Aggiornamenti: i documenti possono essere eliminati e aggiunti di nuovo ma non aggiornati.

- Ricerca, che avviene sui termini anziché su stringhe e può quindi, ad esempio, ignorare le differenze tra singolari e plurali a seconda di come sono impostati i filtri di analisi del testo.

- Risultati ordinati per rilevanza rispetto alla query.

- Commit lenti, in quanto ogni aggiornamento dell'indice richiede la ricostruzione delle memorie di cache, cosa che può richiedere da pochi secondi ad alcuni minuti, a seconda della configurazione.

8.3 Configurazione

La configurazione di Solr comprende la definizione dello schema dati e le impostazioni delle funzionalità della servlet stessa.

Lo schema, definito in `schema.xml`, contiene le informazioni sulla strutturazione dei dati in Solr. Ogni documento indicizzato sarà formato da una serie di "campi statici" (*static fields*), definiti appunto nello schema, e eventualmente da una serie di "campi dinamici" (*dynamic fields*) definiti in al volo. Ogni campo possiede un nome che lo identifica univocamente e un tipo (es: numero, testo, data ecc...); anche i tipi devono essere definiti nel file dello schema, e per ognuno di essi si possono impostare dei filtri che analizzano il testo da indicizzare con estrema flessibilità. Ad esempio, Solr mette a disposizione filtri per spezzare le frasi in parole, rimuovere tag HTML, sostituire i termini con loro sinonimi ecc... Il file `schema.xml`, è diviso in sezioni racchiuse nei seguenti tag:

- `<types>` : contiene le dichiarazioni dei tipi di dati (es. numeri, date, testo, valori booleani) memorizzati nei campi dello schema, a cui si fa riferimento poi nella sezione `<fields>`. Una dichiarazione di tipo booleano può essere:

```
<fieldType name="boolean" class="solr.BoolField"
           sortMissingLast="true" omitNorms="true"/>
```

Un tipo di campo possiede un nome univoco ed è implementato da una classe di Java specificata nell'attributo `class`. Per ogni tipo possono essere impostate varie opzioni; le più importanti sono così definite:

- `indexed` indica che il dato del campo deve essere ricercabile o ordinabile. Se `indexed` è `false`, allora `stored` deve essere `true`. I campi sono `indexed` di default, in caso contrario essi vengono inclusi solamente nei risultati delle ricerche. Un campo `indexed` può avere le seguenti proprietà:
  - `sortMissingFirst`, `sortMissingLast`: se si ordinano i risultati secondo un campo con uno di questi due attributi impostato a `true`, allora si indica se mostrare in testa o in coda i documenti che non possiedono dati nel campo specificato. Il comportamento predefinito prevede che i documenti in questione appiano in testa per ordinamento crescente e in coda per ordinamento decrescente.
  - `omitNorms`: impostazione avanzata che permette di risparmiare memoria; da abilitare se la lunghezza del campo non interessa il punteggio di rilevanza o se il campo non viene usato per calcolare tale punteggio dei risultati.
- `stored`: indica che il campo deve essere incluso nei risultati delle ricerche. Se un campo non è `stored`, allora `indexed` deve essere `true`. Di default `stored` è `true`, ma a volte non è così per quei campi usati per mantenere copie dei dati al fine di aumentare le prestazioni delle ricerche.
  - `compressed`: indica che i dati del campo `stored` devono essere compressi per ridurre lo spazio su disco. Solo i campi con classi `StrFiled` o `TextField` sono comprimibili.
- `multiValued`: se impostato a `true`, il campo può contenere più di un valore. L'ordine è quello definito al momento dell'indicizzazione.

- `<fields>` : contiene le definizioni dei campi dello schema dati. I campi statici sono identificati dal tag `<field/>`, quelli dinamici da `<dynamicField/>`. Un campo dinamico ha la peculiarità di essere creato al volo al momento dell'indicizzazione, in base al pattern specificato nell'attributo `name`. In ogni campo si possono specificare di nuovo, sovrascrivendole, le stesse opzioni per i tipi; inoltre esso possiede anche i seguenti attributi:
  - `name`: l'identificatore univoco del campo.

- `type`: il tipo di campo, come specificato in precedenza nello schema.
- `default`: (opzionale) il valore predefinito del campo. Questo parametro è spesso usato per campi che registrano la data e l'ora di indicizzazione di un documento, per cui il valore predefinito è impostato a `NOW`.
- `required`: (opzionale) se impostato a `true` indica a Solr di dare un messaggio di errore nel caso il documento da indicizzare non abbia un valore per il campo.

- `<copyField/>`: direttive identificate da questo tag vanno inserite dopo e all'esterno del tag `fields` e, al momento dell'indicizzazione di un nuovo documento, indicano a Solr di copiare il contenuto del campo specificato nell'attributo `source` nel campo di destinazione specificato in `dest`. Questa funzionalità è utile quando un valore deve essere salvato in campi designati per diversi scopi, come l'ordinamento, la visualizzazione delle faccette o per migliorare le prestazioni della ricerca.

- Altri tag importanti poi sono: `<uniqueKey>` che specifica quale campo sia l'analogo della chiave primaria delle basi di dati; `<defaultSearchField>` che indica su quale campo effettuare la ricerca, se non specificato nella query; `<solrQueryParser>` che permette di specificare l'operatore di default per lo schema (AND oppure OR).

La configurazione della servlet avviene invece mediante il file `solrconfig.xml`. Informazioni di particolare interesse riguardano i *request handler*, definiti negli omonimi tag `<requestHandler>`. Ogni request hadler viene associato ad un URL specifico, definito nell'attributo `name`, e quindi tutte le query inviate a quell'URL passeranno attraverso quel request handler. Un request handler non è altro che una configurazione di default dei parametri di una query. Ad esempio,

```
<requestHandler name="standard" class="solr.SearchHandler"
                                default="true">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
        <int name="rows">10</int>
        <str name="fl">*</str>
        <str name="version">2.1</str>
    </lst>
</requestHandler>
```

configura il request handler di default impostando i parametri `echoParams`, `rows`, `fl` e `version` evitando così che le applicazioni debbano specificarli ad ogni interrogazione.

8.4 Indicizzazione e reperimento dei dati

L'indicizzazione dei documenti in Solr può avvenire tramite i seguenti meccanismi:

- XML riconosciuto nativamente: Solr riconosce una specifica sintassi di XML che specifica i documenti e i loro campi da indicizzare
- *Character Separated Value* (CSV): i dati vengono inviati alla servlet sotto forma di valori separati da un carattere, solitamente la virgola.
- Importazione diretta da database e da XML: Solr possiede un'estensione chiamata *DataImportHandler* (DIH) che può comunicare sia con le basi di dati sia con sorgenti di XML (es. i servizi web) e supporta varie opzioni di configurazione per associare i campi e modificarne i valori a seconda delle esigenze prima dell'indicizzazione.
- Importazione di documenti complessi attraverso Solr Cell: grazie a questa estensione è possibile importare il testo riconosciuto all'interno di documenti complessi quali PDF, Microsoft Office o persino immagini e file audio.

Effettuare una ricerca, poi, è molto semplice: l'operazione consiste nel chiamare l'URL di un request handler di Solr specificando nella *query string* i parametri (in formato *URL-Encode*). Ad esempio l'URL

```
http://localhost:8983/solr/select?indent=on&version=2.2&q=*%3A*&start
=0&rows=10&fl=*%2Cscore&qt=standard&wt=standard&explainOther=&hl.fl=
```

è così composto:

- `/solr/` è l'indirizzo della servlet di Solr installata nel server.
- `/select` è il riferimento al relativo request handler.
- Seguono infine, dopo il `?`, una serie di parametri in formato `nome=valore` separati dal carattere `&` che indicano le opzioni di ricerca.

Il formato predefinito dei risultati è l'XML, ma la query può essere configurata tramite il parametro `wt` per mostrare i dati in formati riconosciuti nativamente da PHP, Java, Python e Ruby. Un parametro utile in fase di debug è `echoParams` che, se impostato a true, include nel responso dell'interrogazione una lista contenente i parametri passati tramite l'URL.

8.5 Solr e la ricerca a faccette

Solr permette di realizzare un sistema di ricerca multidimensionale: i campi dello schema costituiscono le faccette e, di conseguenza, i termini indicizzati per ogni campo sono i foci. Grazie alle funzionalità di conteggio e alla capacità di filtrare le query in maniera sequenziale, Solr è in grado di fornire, di raffinamento in raffinamento, un contatore per ogni termine indicizzato, rilevando così il numero totale di documenti che lo contengono e guidando la navigazione verso il sottoinsieme di risultati più di interesse per l'utente.

Solr genera questi contatori contestualmente ai risultati della ricerca testuale e dunque entrambe le informazioni vengono restituite a seguito di un'unica interrogazione; in SQL invece un risultato simile potrebbe richiedere una serie di query separate.

Solr prevede tre modalità di funzionamento:

- Conteggio dei valori testuali dei campi: è la modalità fondamentale e più comune di conteggio che interessa tutti i tipi di dati ad esclusione delle date e dei numeri.
- Conteggio delle date: permette di contare documenti che rientrano in specifici intervalli temporali.
- Conteggio delle query: conta i documenti che soddisfano ogni query specificata; è la modalità usata per intervalli numerici.

Ognuna di queste modalità è impostabile da alcuni parametri specifici, ma, in ogni caso, il calcolo dei contatori va abilitato mediante il parametro `facet` impostato a `true` o `on`, altrimenti tutte le altre opzioni sulle faccette vengono ignorate.

8.5.1 Conteggio dei valori testuali

Seguono i parametri tipici per i contatori di faccette testuali:

- `facet.field`: deve essere impostato al nome di un campo su cui attivare i contatori. Questo parametro va ripetuto per ogni campo desiderato.

Poi, i seguenti parametri vanno riferiti ad ogni faccetta ed impostati usando la sintassi come nell'esempio: `f.nomeCampo.facet.sort=lex` (dove `facet.sort` è appunto il parametro della faccetta `nomeCampo`).

- `facet.sort`: impostato a `count` ordina i termini secondo numero di occorrenze decrescente, impostato a `lex` li ordina alfabeticamente.

- `facet.limit`: limita il numero massimo di termini visualizzato per il campo.
- `facet.offset`: è il numero di indice nell'elenco dei termini visualizzabili, che permette di dividere i foci in pagine in caso essi siano numerosi.
- `facet.mincount`: solo i termini con un contatore maggiore a questo numero vengono visualizzati nei risultati.
- `facet.missing`: se impostato a `true` include un contatore "anonimo" riferito a tutti quei documenti che non possiedono alcun termine indicizzato per il campo in questione.
- `facet.prefix`: visualizza contatori solo per termini che iniziano con questa stringa.

8.5.2 Conteggio delle date

Solr supporta nativamente il conteggio su campi che contengono date, secondo intervalli temporali definiti dall'utente. I parametri richiesti a questo scopo sono i seguenti:

- `facet.date`: deve essere impostato, se necessario più volte, al nome dei campi per cui si vuole attivare questa funzionalità.

Il resto dei parametri va specificato con riferimento a ogni campo interessato dal conteggio, come nel caso dei valori testuali; per esempio: `f.nomeCampo.facet.date.start`.

- `facet.date.start`: obbligatorio, indica l'inizio dell'intervallo temporale secondo la sintassi *Date Math*[5]
- `facet.date.end`: obbligatorio, indica il limite minimo per la fine dell'intervallo temporale, sempre secondo la sintassi Date Math.
- `facet.date.gap`: obbligatorio, indica la lunghezza dei sotto intervalli in cui dividere l'intervallo `facet.date.end - facet.date.start`. Il valore da impostare è una durata temporale, non una data precisa, e deve sempre iniziare con un +. Esempi: `+1YEAR` divide l'intervallo in anni, oppure `+1MINUTE+30SECONDS`.
- `facet.date.hardend`: default `false`. Indica a Solr cosa fare se `facet.date.gap` non divide perfettamente l'intervallo indicato. Se impostato a `true` allora l'ultimo sotto intervallo avrà una durata inferiore degli altri, altrimenti il parametro

[5] cfr. http://wiki.apache.org/solr/SolrQuerySyntax

`facet.date.end` viene prolungato in modo tale che tutti i sotto intervalli risultino uguali.

- `.date.other`: default `none`. Questo parametro aggiunge altri contatori a seconda del valore impostato; esso può essere specificato più volte.
  - `before`: conta i documenti il cui valore del campo indica una data che precede il range specificato.
  - `after`: conta i documenti il cui valore del campo indica una data che è posteriore al range specificato.
  - `between`: conta i documenti nel range specificato; opzione ridondante.
  - `none`: disabilitato.
  - `all`: equivalente a tutte tre le opzioni (`before`, `between` e `after`).

8.5.3 Conteggio su query

In questa modalità, invece di scegliere i campi su cui attivare i contatori, si specificano delle query che fungono da faccette. Per ognuna delle query specificate viene contato il numero di documenti trovati, nell'insieme comunque dei risultati della ricerca testuale, e tale valore viene riportato nel responso.

L'unico parametro di configurazione del conteggio su query è `facet.query`, che va impostato ad una qualsiasi query che rispetti la sintassi di Solr. Specificando questo parametro più volte si possono valutare più query contemporaneamente. Il conteggio su query è l'unico modo per generare dei contatori su intervalli numerici.

Pugh riporta un esempio tratto dal sito di informazioni musicali MusicBrainz: nello schema che contiene informazioni su canzoni, i campi `t_name` e `t_duration` indicano rispettivamente il titolo e la durata in secondi delle tracce indicizzate. Il seguente responso di un'interrogazione a Solr chiarifica il meccanismo di conteggio su query; notare l'uso di `echoParams` per includere nell'XML di risposta i parametri usati.

```
<response>
   <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">106</int>
      <lst name="params">
         <str name="indent">on</str>
         <str name="rows">0</str>
         <str name="q">t_name:Geek</str>
         <arr name="facet.query">
```

```
            <str>t_duration:[* TO 119]</str>
            <str>t_duration:[120 TO 179]</str>
            <str>t_duration:[180 TO 239]</str>
            <str>t_duration:[240 TO *]</str>
         </arr>
         <str name="facet">true</str>
      </lst>
   </lst>
<result name="response" numFound="200" start="0"/>
   <lst name="facet_counts">
      <lst name="facet_queries">
         <int name="t_duration:[* TO 119]">55</int>
         <int name="t_duration:[120 TO 179]">36</int>
         <int name="t_duration:[180 TO 239]">64</int>
         <int name="t_duration:[240 TO *]">45</int>
      </lst>
      <lst name="facet_fields"/>
      <lst name="facet_dates"/>
   </lst>
</response>
```

In questo esempio, `facet.query` è stato impostato quattro volte in modo da dividere i documenti, che soddisfano la query `t_name:Geek`, in quattro gruppi secondo il valore del parametro `t_duration`: meno di 2 minuti, da 2 a 3 minuti, da 3 a 4 minuti e più di 4 minuti. La somma dei contatori è 200, che è appunto il totale di documenti trovati, come indicato anche dall'attributo `numFound`. Le query su cui attivare i contatori possono essere di qualsiasi tipo, riferirsi a più campi e non dare necessariamente luogo a insiemi disgiunti, come accade invece nell'esempio.

9 La piattaforma TYPO3

9.1 Caratteristiche generali

TYPO3[34] è un Content Management System (CMS) per il web, scritto in linguaggio PHP utilizzando il paradigma di programmazione ad oggetti. Grazie alla sua modularità, questo software risulta estremamente flessibile. Alcune delle caratteristiche più importanti di TYPO3 sono:

- Universalità: le necessità degli sviluppatori sono ampiamente coperte da una vasta gamma di estensioni che aggiungono al nucleo della piattaforma un gran numero di funzionalità (es: gallerie di immagini, mailing list, e-commerce, supporto multilingue, ecc...) che condividono la medesima base di dati relazionale.

- Gestione degli utenti: TYPO3 possiede un potente sistema di gestione degli account utente; permessi e diritti di accesso possono essere assegnati ad ogni componente del sistema ed ogni categoria di utenti può essere configurata per svolgere solo i propri compiti specifici.

- Gerarchia delle pagine: le pagine sono organizzate in una gerarchia ad albero molto flessibile in cui i documenti possono essere raggruppati o riutilizzati in più rami grazie a link e punti di innesto (*mount points*).

- Tipizzazione dei contenuti: TYPO3 possiede una serie di tipologie di contenuti già configurati per una corretta visualizzazione nel browser. Per esempio è possibile creare del semplice testo, oppure del testo con immagine: nel secondo caso l'immagine può aprirsi in una nuova finestra quando l'utente ci clicca sopra. Chi crea i contenuti non ha bisogno di preoccuparsi di come funziona il codice che realizza questo comportamento: è sufficiente impostare un'opzione nell'editor visuale. Tabelle, moduli e contenuti multimediali possono essere facilmente incorporati nelle pagine senza conoscere i meccanismi che ne stanno alla base.

- Utilizzo di memorie di cache per velocizzare l'output.

9.2 Backend e Frontend

TYPO3 possiede un pannello di amministrazione (*backend*) suddiviso in tre aree principali: i moduli dei menù, la struttura delle pagine ed il modulo dell'area di lavoro. Nel piccolo frame superiore sono inoltre presenti opzioni aggiuntive, quali anteprima delle pagine e controllo della cache, mentre il frame inferiore ospita collegamenti veloci definiti dall'utente. L'interfaccia backend rappresenta il cuore dell'intero Content Management System, in essa, oltre alle operazioni di sviluppo, configurazione ed amministrazione del portale web, si svolgono le fasi di creazione e pubblicazione delle informazioni da parte dello staff editoriale.

L'utente fruitore dei servizi e dei contenuti della piattaforma avrà accesso alla sola interfaccia di navigazione, detta *frontend*, il portale web vero e proprio, che gestirà le operazioni di interazione con l'utente (navigazione, inserimento di dati, la loro rielaborazione, ecc...).

10 Integrazione TYPO3 e Solr

In questo capitolo si presenterà il resoconto dell'attività di integrazione effettivamente svolta, al fine di illustrare l'uso di Solr per applicare la ricerca a faccette alle pagine di un sito Internet realizzato con TYPO3. I seguenti paragrafi tratteranno l'installazione e la configurazione di TYPO3, Tomcat e Solr su sistema operativo Microsoft Windows.

10.1 Installazione del software

TYPO3 richiede l'uso di un DBMS[6], come ad esempio MySQL, nonché di un server web in grado di interpretare il linguaggio PHP, tramite un apposito plug-in.

L'integrazione di queste tre componenti a fini di test è resa semplice grazie a un software di installazione guidata chiamato TYPO3Winstaller [35]. Questo tool in primo luogo configura il web server Apache HTTP Server 2 con PHP 5 e MySQL 4, poi permette la creazione di una o più istanze di TYPO3 con pochi clic del mouse. Fatto ciò è possibile accedere al backend cliccando su "Start TYPO3" nel pannello di controllo dell'installer.

Per installare l'estensione "Apache Solr for TYPO3" nell'istanza di prova è sufficiente recarsi nella sezione "Ext Manager" del backend di TYPO3, selezionare "Import Extensions" dal menu a tendina in alto e immettere "solr" nel campo di ricerca "Look up extensions". Quindi si potrà procedere con l'installazione dell'estensione, e di eventuali altre estensioni da cui essa dipende.

L'installazione di Tomcat come Windows Service è anch'essa semplice, grazie all'installer dedicato. Di default l'accesso al pannello di amministrazione di Tomcat è disabilitato, quindi, terminata la procedura guidata, è necessario modificare il file `tomcat-root\conf\tomcat-users.xml` e aggiungere la seguente stringa dentro il tag `<tomcat-users>`:

```
<user username="admin" password="admin" roles="admin,manager"/>
```

A questo punto, collegandosi all'indirizzo `http://localhost:8080/manager/html` è possibile accedere e gestire le varie servlet installate.

Per installare Solr e configurare lo schema per l'estensione di TYPO3 sono necessari i seguenti passi:

- Scaricare l'archivio contenente Solr e decomprimerlo in una cartella temporanea (`solr-tmp`).

[6] Database Management System, un sistema software per la gestione di basi di dati

- Copiare il contenuto di `solr-tmp\example\solr` in una nuova cartella (per brevità il percorso completo sarà qui identificato da `tomcat-root\solr`) che ospiterà dati e file di configurazione della servlet.
- Copiare `solr-temp\dist\apache-solr-x.y.z.war` in `tomcat-root\webapps\solr.war` (dove `x.y.z` è il numero di versione).
- Dalla sottocartella dell'estensione di TYPO3, `typo3conf\ext\solr\resources\solr\` copiare i file `schema.xml` e `solrconfig.xml` nella cartella `tomcat-root\solr\conf`
- Nel file `tomcat-root\solr\conf\solrconfig.xml` commentare la riga: `<dataDir>/var/lib/solr/data</dataDir>` e, a soli fini di test, dentro il tag `autoCommit` impostare `maxDocs` a `1`.
- Modificare il file `tomcat-root\solr\conf\server.xml` e aggiungere l'attributo `URIEncoding="UTF-8"` al tag `<Connector />`.
- Creare il file `tomcat-root\conf\Catalina\localhost\solr.xml` con il contenuto:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context docBase="tomcat-root\webapps\solr.war" debug="0"
   crossContext="true">
    <Environment name="solr/home" type="java.lang.String"
       value="tomcat-root\solr" override="true" />
</Context>
```

- Infine riavviare Tomcat. Il pannello di controllo di Solr è ora accessibile all'indirizzo `http://localhost:8080/solr/admin`.

10.2 Configurazione di TYPO3

La configurazione di TYPO3 è avvenuta attraverso i template e TypoScript, il linguaggio di programmazione proprio della piattaforma, mentre l'aspetto visivo del frontend è stato affidato all'estensione TemplaVoilà, che semplifica lo sviluppo dei temi grafici.

10.2.1 Configurazione e parametri di connessione

Per il corretto funzionamento dell'interfaccia con Solr, è necessario includere nel template in uso i plug-in "Apache Solr" (`solr`), "CSS Styled Content" (`css_styled_content`)

e “Page Browser” (`pagebrowse`). Inoltre è necessario definire le seguenti costanti, indicando i parametri del server Tomcat in cui è installato Solr (indirizzo IP, porta e percorso della servlet).

```
plugin.tx_solr.solr.host = 127.0.0.1
plugin.tx_solr.solr.port = 8080
plugin.tx_solr.solr.path = /solr/
```

10.2.2 Parametri per l’indicizzazione

Per permettere a TYPO3 di indicizzare le pagine in Solr è necessario quindi:

- Impostare l’opzione “Is root of website” (`is_siteroot`), nella pagina principale del sito.
- Abilitare l’indicizzazione, inserendo nel TypoScript del template l’istruzione

  ```
  config.index_enable = 1
  ```

- Nella pagine visualizzate nel frontend assicurarsi che i dati da indicizzare siano racchiusi tra i tag `<!-- TYPO3SEARCH_begin -->` e `<!-- TYPO3SEARCH_end -->`.

Già così, senza alcuna impostazione aggiuntiva, ogni pagina visitata dal frontend viene indicizzata in Solr. L’estensione si preoccupa, pagina per pagina, di specificare il contenuto dei vari campi definiti in `schema.xml`, come ID della pagina, URL, autore, data di indicizzazione, contenuto, ecc...

Inoltre, l’estensione Apache Solr for TYPO3 rende molto semplice la creazione di nuovi campi personalizzati senza dover mai modificare il file dello schema: ciò è possibile grazie ai campi dinamici, definibili tramite TypoScript, nel template di ogni pagina, tramite l’istruzione

```
plugin.tx_solr.index.additionalFields.NomeCampo = Valore
```

dove `NomeCampo` è il nome del campo dinamico che si vuole creare e `Valore` è il contenuto che si vuole attribuire al campo per la pagina in questione. Il nome del campo dinamico deve comunque soddisfare uno dei pattern specificati in `schema.xml`[7].

La definizione dei campi personalizzati supporta i cObjets, quindi è stato possibile definire i vari campi una sola volta nel template principale, grazie alle istruzioni

[7] Per i dettagli sui campi dinamici vedi http://wiki.apache.org/solr/SchemaXml#Dynamic_fields

```
plugin.tx_solr.index.additionalFields.NomeCampo = TEXT
plugin.tx_solr.index.additionalFields.NomeCampo.field = nome_campo_pagina
```

Così facendo, si impone che `NomeCampo` sia un oggetto `TEXT`, il quale può leggere il proprio valore testuale da uno dei campi di input (`nome_campo_pagina`) presenti nel pannello delle proprietà di ogni pagina e creati ad hoc con l'estensione KickStarter. In questo modo non solo si è risparmiato l'onere di ridefinire i campi dinamici e il loro contenuto pagina per pagina, ma anche, grazie all'ausilio dei campi di input personalizzati, si è reso il processo di creazione e definizione delle faccette molto più user-friendly rispetto alla scrittura di codice.

10.2.3 Il campo di ricerca e la visualizzazione dei risultati

L'estensione Solr for TYPO3 mette a disposizione un plug-in per il frontend che visualizza un campo di ricerca e gli eventuali risultati. Questo plug-in può essere attivato in due modi: inserendolo come record tra i contenuti di ogni pagina in cui si vuole rendere disponibile la ricerca, oppure visualizzando, via TypoScript, l'output dell'oggetto `plugin.tx_solr_pi_results`. In questo secondo caso, il template da modificare è solo quello della pagina principale, il quale poi verrà ereditato da tutte le sottopagine. Impostazioni rilevanti per il campo di ricerca sono:

- `plugin.tx_solr.search.targetPage` va impostato all'ID numerico della pagina in cui verranno visualizzati i risultati della ricerca; anche la pagina di destinazione dovrà ospitare il plug-in "Solr Search", che, per il corretto funzionamento, dovrà avere come `targetPage` il medesimo ID.

- `plugin.tx_solr.search.allowEmptyQuery` va inpostato a `1` o `0` a seconda che si voglia abilitare o meno la ricerca con query vuota, che restituirà tutti i documenti indicizzati.

Per visualizzare le informazioni sulle faccette invece è necessario modificare il template della pagina dei risultati. Le impostazioni di interesse a riguardo sono:

- `plugin.tx_solr.search.faceting` va impostato a `1` per abilitare i contatori sui termini dei campi.

- `plugin.tx_solr.search.faceting.minimumCount` va impostato al valore minimo che ogni contatore deve avere per poter essere visualizzato. Solo i termini contati un tale numero sufficiente di volte verranno mostrati nei risultati.

- `plugin.tx_solr.search.faceting.facets` è un array di oggetti che indica tutti i campi dello schema (dinamici e non) su cui attivare i contatori e il loro comportamento, ad esempio una semplice faccetta testuale viene impostata in questo modo:

```
plugin.tx_solr.search.faceting.facets {
        nome_faccetta1 {
                field = nome_campo
                label = Nome del campo
        }
        nome_faccetta2 {
                ...
        }
}
```

  - `nome_faccetta1` è un nome fittizio dato a piacere all'oggetto dell'array;
  - `field` è il riferimento al campo dello schema su cui attivare i contatori;
  - `label` è il titolo del campo che verrà visualizzato nel frontend.

Per impostazioni avanzate, come abilitare la selezione multipla di più termini tramite check-box, evidenziazione dei risultati, ordinamento delle faccette, quali faccette visualizzare, quali nascondere e altro ancora si rimanda alla documentazione ufficiale [36].

L'aspetto del modulo di ricerca e dei risultati è definito nel template presente nel file: `typo3conf\ext\solr\resources\templates\pi_results\results.htm`

10.3 Considerazioni e sviluppi futuri

Apache Solr for TYPO3 si prefigge l'obiettivo di integrare un sitema di gestione dei contenuti complesso ma flessibile, con un sistema di ricerca ricco di aspetti innovativi. La versione 1.1.0 esaminata in questo studio, però, riesce a coprire solo parzialmente le funzionalità offerte da Solr, a tal punto che non si ritiene questa estensione un software sufficientemente maturo per andare incontro agli sviluppatori più esigenti, che cercano di sfruttare appieno le potenzialità del software del gruppo Apache.

I limiti, riscontrati durante il test e la messa in opera di un sistema di prova, riguardano i seguenti aspetti:

- L'aggiunta di campi personalizzati (ovvero le faccette) va fatta solo via codice: sarebbe gradito, e renderebbe più agevole l'uso, un pannello, un'interfaccia grafica, che aiuti a definire i campi dinamici a livello globale o, ancora meglio, secondo gruppi di pagine.

- L'aggiunta di campi di input personalizzati per le proprietà di ogni pagina, che andranno ad assegnare uno o più valori alle faccette citate al punto precedente, è lasciata a estensioni terze. In questo studio si è utilizzata l'estensione KickStarter a tal fine, ma si ritiene che ciò debba essere realizzato con il minimo lavoro da parte del webmaster.

- Manca il supporto per le faccette gerarchiche, ovvero organizzate ad albero: la versione 1.1.0 non implementa quella che è ritenuta una delle funzionalità più utili e interessanti della ricerca a faccette.

- Manca il supporto per contatori su date su query arbitrarie: nel codice sorgente le funzioni che dovrebbero implementare questi aspetti fondamentali contengono solo il commento "*TO DO*".

- Le pagine vengono indicizzate solo al momento che un utente le visita dal frontend; è necessario un crawler, le modifiche non appaiono subito nei risultati della ricerca e i documenti vanno eliminati manualmente dallo schema di Solr.

- La compatibilità con le versioni più recenti di PHP (dalla 5.3 in poi) per i sistemi operativi Windows è da migliorare.

Visitando il *bugtracker*[8] del progetto si scopre che molti di questi problemi sono stati presi in considerazione o sono stati risolti da parte del team di sviluppo, che prevede di rilasciare al pubblico la versione 2.0 all'inizio del 2011 [37]. Inoltre, esiste una versione *closed source* dell'estensione che rende disponibili via TypoScript le funzioni più avanzate come la definizione di faccette gerarchice, i suggerimenti per la ricerca e l'autocompletamento delle query. Nel sito degli sviluppatori è disponibile una tabella comparativa tra la versione open source e quella a pagamento [38].

[8] http://forge.typo3.org/projects/extension-solr/issues

11 Conclusioni

Questo studio ha presentato una tecnica per l'accesso alle informazioni le cui origini teoriche risalgono ai primi decenni del Novecento. Tuttavia la sua applicazione in ambito informatico è stata concretizzata solo recentemente.

Come si è visto, anche se il problema della classificazione ha interessato la nostra cultura per millenni, le idee più rivoluzionare sono state formulate nel ventesimo secolo e dal futuro non ci si può aspettare altro che ulteriori evoluzioni.

Alcuni ricercatori si sono già imbattuti nei limiti espressivi della classificazione a facette e si sono spinti oltre studiando le cosiddette "ontologie". In greco la parola ontologia significa "studio dell'essere" e, nell'ambito della scienza dell'informazione, indica la rappresentazione di insiemi di oggetti e delle relazioni che intercorrono tra essi [39].

Mentre questi nuovi approcci alla conoscenza sono ancora in fase di studio, si è visto come la ricerca a faccette abbia già preso piede nel web grazie anche alle implementazioni open source. D'altronde, con lo sviluppo di sistemi destinati a memorizzare enormi moli di dati, la ricerca a faccette viene incontro alla necessità di rendere tali dati agevolmente disponibili ad un pubblico di utenti potenzialmente inesperti, che riescono così a reperire facilmente le informazioni senza sapere a priori come trovarle.

Dunque ci si aspetta che di qui in avanti gli sviluppatori di tutto il mondo continueranno ad affinare i software e i design pattern per cercare di sfruttare il più possibile le potenzialità di un approccio alla ricerca che esalta la quantità di informazioni senza compromettere la qualità dell'esperienza utente.

Riferimenti bibliografici

[1] Reitz, J. 2007. *Online Dictionary for Library and Information Science.* http://lu.com/odlis/

[2] Wikipedia: *Exploratory Search.* http://en.wikipedia.org/wiki/Exploratory_search

[3] Aristotele. 343 AC. *Historia Animalium.*

[4] Aristotele. 350 AC. *De Partibus Animalium.*

[5] Aristotele. 350 BC. *De Generatione Animalium.*

[6] Linneo, C. 1767. *Systema Naturae.*

[7] Gabrielli, A. 2008. *Grande Dizionario Italiano.* Milano: Hoepli

[8] Dewey, M. 1876. *A Classification and Subject Index for Cataloguing and Arranging the Books and Pamphlets of a Library.* Project Gutenberg. http://www.gutenberg.org/etext/12513

[9] http://www.google.it/dirhp

[10] http://dir.yahoo.com/

[11] http://www.dmoz.org/

[12] OCLC Online Computer Library Center. *Introduction to Dewey Decimal Classification.* http://www.oclc.org/dewey/versions/ddc22print/intro.pdf

[13] Ranganathan, S. R. 1933. *Colon Classification.* Madras, India: Madras Library Association.

[14] Ranganathan, S. R. 1950. *Classification, Coding, and Machinery for Search.* Parigi: UNESCO. http://unesdoc.unesco.org/images/0013/001333/133325eo.pdf

[15] Wikipedia: *Standard Boolean Model.* http://en.wikipedia.org/wiki/Standard_Boolean_model

[16] Manning, C., Raghavan, P., and Schütze, H. 2008. *Introduction to Information Retrieval.* New York: Cambridge University Press. http://nlp.stanford.edu/IR-book/

[17] Wictionary. *meta- http://en.wiktionary.org/wiki/meta-*

[18] Wikipedia. *Metadato*. http://it.wikipedia.org/wiki/Metadato

[19] The Bliss Classification Association. http://www.blissclassification.org.uk

[20] Broughton, V. 2001. *A classification for the 21st century: principles and structure of the Bliss Bibliographic Classification*. Vjesnik Bibliotekara Hrvatske 44(1-4). Traduzione: http://www.aib.it/aib/contr/broughton1.htm

[21] Ranganathan, S. R. 1967. *Prolegomena to Library Classification*

[22] Spiteri, L. 1988. *A Simplified Model for Facet Analysis*. Canadian Journal of Information and Library Science v23, 1-30. http://iainstitute.org/en/learn/research/a_simplified_model_for_facet_analysis.php

[23] Vickery, B. C. 1960. *Faceted classification: a guide to construction and use of special schemes*. London: Aslib.

[24] Denton, W. 2003. *How to Make a Faceted Classification and Put It On the Web*. http://www.miskatonic.org/library/facet-web-howto.html

[25] Kwasnick, B. H. 1999. *The role of classification in knowledge representation and discovery*. Library Trends 48 (1): 22-47.

[26] Grimes S. 2006. *Unstructured Data and the 80 Percent Rule* http://www.clarabridge.com/default.aspx?tabid=137&ModuleID=635&ArticleID=551

[27] Tunkelang, D. 2009. *Faceted Search*. pp52-53. Morgan Claypool.

[28] Wikipedia. *AJAX*. http://it.wikipedia.org/wiki/AJAX

[29] Adobe Systems Inc. http://www.adobe.com

[30] Site analytics for amazon.com http://siteanalytics.compete.com/amazon.com/

[31] Smiley, D., Pugh E. 2009, *Solr 1.4 Enterprie Search Server*. Birmingham: Packt Publishing.

[32] Apache Tomcat. http://tomcat.apache.org/

[33] Jetty WebServer. http://jetty.codehaus.org/jetty/

[34] TYPO3. http://typo3.org/

[35] TYPO3Winstaller. http://typo3winstaller.sourceforge.net/

[36] Apache Solr for TYPO3. Wiki. http://forge.typo3.org/projects/extension-solr/wiki

[37] Apache Solr for TYPO3. Roadmap. http://www.typo3-solr.com/en/solr-for-typo3/roadmap/

[38] Apache Solr for TYPO3. Extension Comparison. http://www.typo3-solr.com/en/solr-for-typo3/development-model/#c1183

[39] Wikipedia. *Ontology (information science).* http://en.wikipedia.org/wiki/Ontology_%28information_science%29