Alma Mater Studiorum - Università di Bologna

Facoltà di Ingegneria
Tesi di Laurea Specialistica in Ingegneria Informatica

Ricerca e Raccomandazioni di Articoli Scientifici in Reti Peer-to-Peer

Leonardo D'Alonzo

Relatore
Chiar.mo Prof. Ing. Paolo Ciaccia

Correlatori
Dr. ir. Johan Pouwelse
Dr. David Hales
Dr. Tamás Vinkó

Anno Accademico 2008/2009

P2P Search and Recommendations of Scientific Literature

Tribler Research Group
Parallel and Distributed Systems Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.pds.ewi.tudelft.nl

P2P Search and Recommendations of Scientific Literature

Abstract

Peer-to-peer (p2p) distributed computing model is based on the sharing of resources between peers which leads to the design and deployment of very large applications with very low cost.

Content location is a general problem in p2p networks. Solutions which rely on broadcast are inherently not scalable, while those based on DHTs lead to high overhead while requiring a keyword-based approach to information retrieval. Most of p2p systems delegates the task of finding relevant content to an information retrieval system based on a client-server paradigm. This approach requires sizeable capital investments in the infrastructure in order to achieve high scalability levels and it is prone to possible data exploitation.

Recent solutions have proposed to organize network topology around data semantics and to exploit the resulting topology in order to facilitate content location. Tribler is a p2p file sharing system which forms a semantic overlay network to identify related content, facilitate search and recommend content to user without the need of a central service.

By relying on this latter model we designed a proof-of-concept fully-decentralized search and recommender system tailored for scientific literature. We proactively manage a social network which aggregates users with similar information needs by relying on a lightweight gossip-based topology management protocol. User's information needs are automatically learned by tracking users reading habits. A statistical model of language, such as the vector space model of text, is used in order to capture the semantics of user's interests. Content location and recommendation are carried out by exploiting the dynamically evolving social network. Relevant information is expected to be located in the user's neighborhood since the network topology has been constructed around user's information needs, i.e. it has been pushed towards relevant information. Interesting paper are recommended by users with similar reading interests.

Preface

This document describes the project I have done for my final exam for the MSc (Laurea Specialistica) in Computer Engineering at University of Bologna. The research was performed at the Parallel and Distributed System Group of the Faculty Electrical Engineering, Mathematics, and Computer Science of Delft University of Technology. I was there from October 2008 to April 2009 as exchange student participating to the LLP Erasmus 2008/2009. I would like to thank Prof.ssa ing. Eleonora Franchi Scarselli who was promoter and coordinator of the exchange.

During those months I worked under supervision of Dr. ir. Johan Pouwelse. He is assistant professor at Delft University of Technology and the scientific director of the P2P-Next European project investigating the next-generation of peer-to-peer technology. I would like to thank him for his contribution, his contagious enthusiasm and for changing my recurring overuse of bottom-up approach to problem solving.

I took advantages in working with Ph.D. Tamás Vinkó, who is expert in global optimization, and Ph.D. David Hales, who is a computational social scientist. I would like to thank them for our perceptive discussions and their pointers.

I would like to thank my supervisor at University of Bologna, Prof. ing. Paolo Ciaccia, for having reviewed this work and for participating in the examination committee.

Finally I would like to thank ir. Frank Mulder for all countless tips he gave me, ir. Riccardo Petrocco and ir. Diego Rabaioli for their technical support.

Leonardo D'Alonzo
Bologna, July 20, 2009

Contents

Ricerca e raccomandazioni di articoli scientifici in reti peer-to-peer

Compendio I

Introduzione

Il modello di computazione distribuito peer-to-peer (p2p) permette la realizzazione a basso costo di servizi attraverso la condivisione delle risorse messe a disposizione da ciascuna delle entità partecipanti (peer).

Un problema di carattere generale in questo tipo di reti è quello della localizzazione dei contenuti, che sono visti come risorse messe a disposizione da ciascun nodo. La maggiore difficoltà nel localizzare contenuti in maniera completamente distribuita è legata al costo richiesto nel fornire una visione globale della rete a ciascuna località. Un approccio alternativo consiste nell'organizzare la topologia della rete in modo che rifletta la semantica dei contenuti messi a disposizione dai singoli nodi e di sfruttare la topologia risultante per facilitare la ricerca. Facendo riferimento a questo ultimo approccio, indagheremo i modelli alla base di un sistema informativo completamente decentralizzato che offra ad utenti accademici un servizio di ricerca e di raccomandazioni di articoli scientifici.

Il paragrafo I.1 introduce il modello di computazione p2p precisandone lo scenario d'uso e di preferibilità al modello client-server. Il paragrafo I.2 introduce le problematiche legate alla localizzazione dei contenuti e dà una panoramica degli approcci di soluzione possibili. Il paragrafo I.3 precisa i requisiti del sistema e introduce l'approccio di soluzione.

I.1 Modello di computazione peer-to-peer

Il modello di computazione peer-to-peer (p2p) si basa sulla interconnessione di un insieme di nodi (*peer*) ciascuno dei quali mette a disposizione una parte delle proprie risorse (e.g. contenuti, potenza computazionale, memoria, banda) per la realizzazione di un servizio. Per contro, il modello client-server partiziona un servizio tra il server, che lo fornisce mettendo a disposizione proprie risorse, e i clienti, che da questo dipendono per poterne usufruire.

Il modello p2p prevede che la computazione avvenga in modo completamente distribuito tra tutti i nodi partecipanti che sono agenti della stessa classe, i.e. ciascuno di essi agisce sia come client che come server. La computazione si basa sull'interazione tra i nodi partecipanti, ciascuno dei quali ha una conoscenza lo-

cale degli altri nodi. Tale conoscenza locale è modellabile con una topologia di interconnessione tra gli agenti partecipanti (*overlay network*) che è indipendente dall'interconnessione, tipicamente IP, delle macchine (*host*) che li ospitano. Tale struttura topologica influisce sulle caratteristiche del sistema, in modo particolare per ciò che concerne la scalabilità [ATS04].

I.1.1 Applicabilità

Anche se un servizio basato sul modello client-server è tecnicamente scalabile (e.g. Google, YouTube, Wikipedia, Facebook), ciò richiede generalmente un considerevole investimento in termini di infrastrutture, che è generalmente possibile solo qualora il servizio sia supportato da un solido modello di business. Inoltre, dato che il servizio è gestito in modo centralizzato, possono emergere problematiche legate alla sicurezza e alla riservatezza dei dati che sono concentrate in un unica entità.

Il modello di computazione p2p induce alla progettazione a basso costo di servizi completamente decentralizzati nei quali non ci siano punti di centralizzazione. I costi legati all'amministrazione, configurazione e sicurezza del servizio sono distribuiti tra tutti gli utenti anziché essere gestiti da un'unica entità. Proprietà desiderabili, quali la scalabilità, la capacità di auto-configurazione e l'adattamento a fronte di guasti e dinamicità della rete senza il supporto di punti di centralizzazione, devono essere tenute in considerazione in fase di progettazione. I sistemi p2p, in particolare modo quelli costituiti su un ampia base dinamica di nodi partecipanti, manifestano proprietà che possono essere descritte con modelli usati tradizionalmente per sistemi biologici o sociali. Un approccio alternativo alla progettazione di sistemi p2p prevede il ricorso a tali modelli [MH08].

I.1.2 Distribuzione di contenuti

Il modello di computazione p2p è stato esaltato dai sistemi di distribuzione di contenuti (e.g. Napster, Gnutella, BitTorrent). Questi sistemi sono costituiti sulla base di un elevato numero di nodi partecipanti tra loro interconnessi a formare una rete, ciascuno dei quali dovrebbe mettere a disposizione del sistema delle risorse, tipicamente contenuti e banda passante. Ciò permette la realizzazione a basso costo di un servizio che avrebbe altrimenti richiesto costi elevati qualora fosse stato realizzato secondo il classico modello client-server.

I.2 Localizzazione di contenuti

Affinché i contenuti a cui un utente è interessato possano essere acceduti, eventualmente con il supporto di un sistema di distribuzione, è necessario che questi vengano prima localizzati. Ciò è generalmente il compito di cui si occupa

un servizio di recupero dell'informazione (IR, i.e. *Information Retrieval*) che indica all'utente i contenuti che possano soddisfare le sue esigenze informative. Servizi di questo tipo richiedono generalmente una visione globale delle informazioni disponibili al fine di selezionare quelle che più si adattano alle esigenze dell'utente, in accordo ad un certo modello di rilevanza. Ciò trova una naturale mappatura nel modello client-server, nel quale il server ha una conoscenza globale delle informazioni messe a disposizione. I più diffusi sistemi di distribuzione di contenuti p2p (e.g. Napster, che ne è stato l'archetipo, ora BitTorrent) risolvono il problema della localizzazione di contenuti delegando questo compito a un servizio di IR basato sul modello client-server.

La maggiore difficoltà nel localizzare contenuti in maniera completamente distribuita, i.e. senza fare affidamento su punti di centralizzazione, è legato al costo richiesto nel fornire una visione globale della rete a ciascuna località. La soluzione più semplice, che è quella di acquisire tale conoscenza contattando tutti i nodi partecipanti, è intrinsecamente non scalabile [CRB+03].

Un servizio pensato secondo il modello client-server può essere completamente distribuito facendo uso del supporto di opportuni middleware che forniscano un servizio di hashing distribuito, i.e. che nascondano la località al nome di una risorsa. Approcci di questo tipo richiedono generalmente un'elevata interazione tra i nodi, in modo particolare qualora si cerchi di distribuire un servizio di IR basato sull'uso di parole chiave per accedere alle informazioni desiderate, tipicamente testuali. Livelli di coordinamento tra i nodi ancora maggiori sono necessari qualora si richiedano proprietà di tolleranza ai guasti ovvero in presenza di alte dinamicità della rete, sia in termini di popolazione dei nodi che di disponibilità delle informazioni. La realizzazione di sistemi di IR che facciano uso di questo approccio mostrano limiti di scalabilità [RV03, LLH+03].

Un approccio alternativo consiste nell'organizzare la topologia della rete di interconnessione in modo che rifletta la semantica dei contenuti messi a disposizione dai singoli nodi e di sfruttare efficientemente la topologia risultante per localizzare le informazioni desiderate [RM06]. Poiché la disponibilità delle informazioni e la popolazione della rete sono supposti dinamici, la topologia di interconnessione dovrebbe evolversi in accordo a tali dinamiche. In situazioni di questo genere, la gestione della topologia può essere affidata a protocolli epidemici che, basandosi su un modello probabilistico della diffusione delle informazioni, sono a bassa intrusione [VvS05]. Tribler[1] è un sistema di distribuzione di contenuti (*file sharing*) basato su BitTorrent che organizza dinamicamente i nodi in una topologia di interconnessione che rifletta la località semantica dei contenuti (SON, i.e. *Semantic Overlay Network*) in modo da facilitarne la ricerca e la raccomandazione agli utenti senza il supporto di punti di centralizzazione che supportino una visione globale della rete [PGW+08]. La topologia della rete emerge sulla base delle ne-

[1] www.tribler.org

cessità informative degli utenti che vengono automaticamente stimate dal sistema osservando la storia dei file che sono stati scaricati.

Il raggruppamento di utenti con simili esigenze informative guida la costruzione di un *social network* nel quale le relazioni tra gli utenti emergono dall'analisi dei dati. Tale approccio può essere considerato un framework sul quale basare lo sviluppo di applicazioni che offrano servizi personalizzati, di filtraggio collaborativo di contenuti e di classificazione ontologica dei dati [AHPE09].

I.3 Contributo

Lo scopo di questo lavoro consiste nell'indagare i modelli alla base di un sistema informativo completamente decentralizzato che offra a utenti accademici un servizio di ricerca e di raccomandazioni di articoli scientifici. Faremo riferimento a soluzioni impiegabili in reti altamente popolate ad elevate caratteristiche di dinamicità sulla base del modello dei diffusi sistemi di file-sharing.

I servizi di ricerca e raccomandazioni possono essere sviluppati facendo uso di un framework di social network nel quale le relazioni tra gli utenti sono basate sulla similarità tra le loro esigenze informative. L'identificazione di utenti con simili interessi si riflette nella struttura topologica della rete di interconnessione che può essere sfruttata per facilitare la ricerca e raccomandare all'utente sia articoli scientifici che utenti con interessi simili [HPK08].

Indagheremo una soluzione basata sul modello adottato dal sistema di file sharing Tribler, adattando il sistema informativo alla specificità dei contenuti, i.e. gli articoli scientifici. Il modello di rilevanza utilizzato per stimare le esigenze informative degli utenti in un sistema di file sharing, può essere ragionevolmente basato sull'analisi dei file scaricati che tuttavia è poco adatto per contenuti testuali. Per questo tipo di contenuti la semantica delle esigenze informative dell'utente può essere dedotta ricorrendo a modelli algebrici del linguaggio. Il modello a spazio vettoriale del testo si basa sull'analisi statistica delle ricorrenze lessicali nel testo al fine si modellarne la semantica. La similarità semantica, tra due documenti testuali è riconducibile al computo di distanze tra vettori di uno spazio vettoriale. Tale modello può essere usato come base per stimare le esigenze informative degli utenti, che possono essere quindi utilizzate per indurre una topologia della rete di interconnessione che rifletta, in base allo stesso modello, similarità nelle esigenze informative. La topologia del social network risultante può quindi essere utilizzata in modo da facilitare la ricerca e raccomandare sia contenuti che utenti.

Compendio II

Un sistema decentralizzato di ricerca e raccomandazioni

Questo capitolo illustra la soluzioni alla base di un servizio completamente decentralizzato di ricerca e raccomandazioni di articoli scientifici. Ciascun utente è supportato da un agente che stima automaticamente le sue esigenze informative analizzandone le abitudini di lettura. Il sistema informativo è basato sull'interazione di una popolazione dinamica di agenti che, organizzandosi in una topologia di interconnessione che riflette le similarità tra le esigenze informative degli utenti, filtra i contenuti resi disponibili in modo da adattarli alle specificità di ciascun utente.

Il paragrafo II.1 illustra l'architettura generale dell'agente informativo e introduce il funzionamento dei moduli componenti. Il paragrafo II.2 descrive il modulo adibito all'estrazione dei dati dagli articoli scientifici. Il paragrafo II.3 discute delle modalità con le quali l'agente stima le esigenze informative dell'utente che sono usate per organizzare una popolazione dinamica di agenti in una topologia di interconnessione. Il protocollo adibito a ciò è descritto nel paragrafo II.4. Il paragrafo II.5 descrive come sfruttare la topologia risultante per facilitare la ricerca e le raccomandazioni di articoli scientifici.

II.1 Architettura del sistema informativo

In figura 0.1 è schematizzata l'architettura dell'agente informativo. Gli articoli scientifici collezionati da ciascun utente sono localmente indicizzati con il supporto di un sistema di estrazione dell'informazione. Questa base di dati è sfruttata dall'agente per stimare le esigenze informative dell'utente, le quali guidano l'organizzazione della popolazione di agenti in una topologia di interconnessione che rifletta le similarità nelle esigenze informative.

Il servizio di ricerca e raccomandazioni di articoli scientifici è realizzato dall'interazione di una popolazione dinamica di agenti, ciascuno dei quali filtra i contenuti resi disponibili dall'intera popolazione in modo da adattarli alle esigenze informative specifiche di ciascun utente.

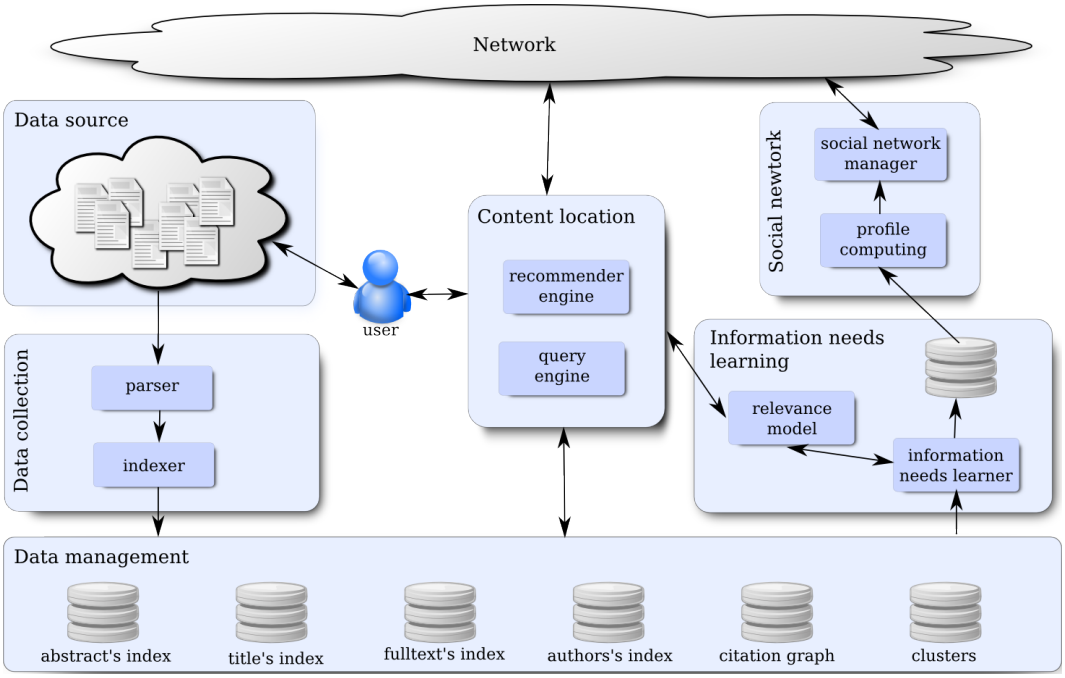


Figure 0.1: Architettura dell'agente informativo e flusso dei dati attraverso i suoi componenti. Dagli articoli scientifici letti dall'utente vengono estratti i metadati, le citazioni e l'intero contenuto testuale, i quali vengono localmente indicizzati. Questa base di dati è sfruttata per stimare le esigenze informative dell'utente, che guidano l'organizzazione degli agenti in una topologia di interconnessione, la quale è sfruttata dal meccanismo di ricerca e raccomandazioni.

II.2 Raccolta dei dati

La nozione di esigenza informativa dell'utente è supportata da un modello che dipende dalle caratteristiche intrinseche degli oggetti informativi, che nel caso specifico sono articoli scientifici. In generale un modello dei dati che catturi molte caratteristiche permette una sintesi più accurata del modello di recupero dell'informazione. Ad esempio, una caratteristica molto importante per gli articoli scientifici sono le citazioni, che costituiscono delle raccomandazioni di lettura fornite dall'autore. Altre peculiarità importanti sono il titolo, gli autori, l'abstract, che caratterizzano ciascun articolo scientifico e dovrebbero essere tenuti in considerazione nella definizione del modello di recupero.

Il mezzo più diffuso per la distribuzione degli articoli scientifici è il formato standard PDF che è un formato orientato alla presentazione, i.e. le informazioni memorizzate sono non strutturate. L'estrazione di informazione strutturata da documenti non strutturati è un problema di difficile formalizzazione e per questo l'approccio migliore fa uso di tecniche di apprendimento auto-

matico [Cha03, MRS08].

Per i nostri scopi abbiamo realizzato un sistema dimostrativo che estrae automaticamente una serie di informazioni strutturate, quali il titolo, l'abstract, l'intero contenuto testuale e le citazioni, da articoli scientifici in formato PDF (cfr. capitolo 2).

II.3 Stima delle esigenze informative

Le necessità informative dell'utente possono essere stimate facendo uso di un modello che tenga in considerazione una serie di misure oggettive riguardanti l'interazione dell'utente con il sistema informativo, come ad esempio l'analisi degli articoli scientifici che sono stati letti, il tempo impiegato nella lettura di ciascuno di essi, le parti sulle quali l'utente si è maggiormente soffermato, l'analisi delle query sottoposte al sistema, etc.

Un modello che dia una stima delle esigenze informative dell'utente a partire da tali segnali è di difficile formalizzazione. In situazioni di questo tipo si adotta un approccio basato su tecniche di apprendimento automatico. Tuttavia, poiché vogliamo indagarne solo le idee fondanti, ci limiteremo a considerare le tecniche che possono essere impiegate nell'analisi di collezioni di articoli scientifici, i quali possono essere trattati alla stregua di pagine Web, i.e. documenti di testo che possono riferirsi tra di loro. Il capitolo 3 illustra in modo più dettagliato le tecniche maggiormente utilizzate nell'analisi di collezioni di questo tipo.

II.3.1 Modello a spazio vettoriale del testo

Il modello a spazio vettoriale del testo caratterizza il contenuto semantico di un documento di testo adottando un modello lessicale del linguaggio (cfr. sezione 3.1). Il vocabolario è trattato come base di uno spazio vettoriale i cui vettori sono combinazioni di vocaboli, i.e. documenti di testo. Ciò è supportato da modelli statistici del linguaggio che cercano di dare un modello della semantica basato sulle occorrenze lessicali dei vocaboli in un testo. Lo schema tf-idf valuta l'importanza di un vocabolo i in un testo j con un peso w_{ij} dipendente da due fattori, uno locale (tf, i.e. *term frequency*), che dipende dal numero delle occorrenze del vocabolo nel testo, e l'altro globale (idf, i.e. *inverse document frequency*) che valuta l'importanza di un vocabolo in base alla sua località nell'intera collezione, e.g. se un termine appare in un solo documento, si suppone che abbia un alto potere caratterizzante.

$$w_{ij} = \mathrm{tf}_{ij} \cdot \mathrm{idf}_i \tag{1}$$

La similarità semantica tra due documenti è valutabile in base alla distanza tra

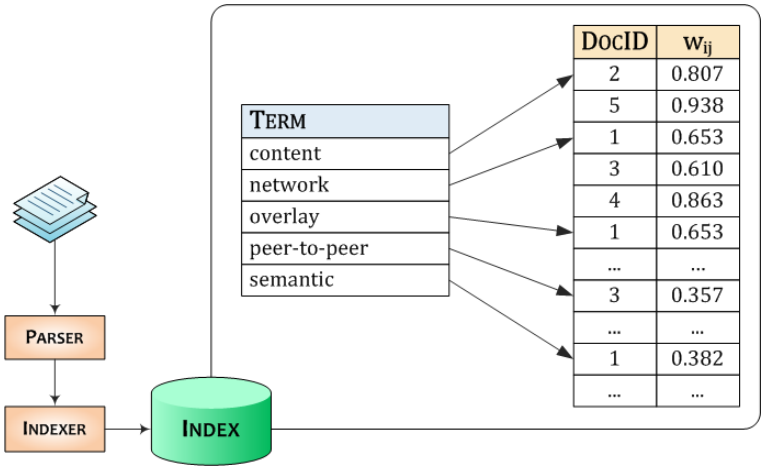


Figure 0.2: Il contenuto testuale degli articoli scientifici è indicizzato localmente adottando il modello a spazio vettoriale del testo. A ciascun vocabolo è associata la lista dei documenti nei quali occorre, unitamente al peso calcolato secondo il modello statistico tf-idf.

i due corrispondenti vettori (normalizzati).

$$sim(\mathbf{doc_j}, \mathbf{doc_k}) = \cos\theta_{j,k} = \frac{\mathbf{doc_j} \cdot \mathbf{doc_k}}{\|\mathbf{doc_j}\|\|\mathbf{doc_k}\|} = \frac{\sum_i w_{i,j} \cdot w_{i,k}}{\sqrt{\sum_i w_{i,j}^2}\sqrt{\sum_i w_{i,k}^2}} \tag{2}$$

Criticità Il modello a spazio vettoriale del testo, poiché modella il contenuto del testo su base lessicale, risente delle ambiguità tipiche del linguaggio naturale, quali ad esempio la molteplicità di significati associabili ad un vocabolo (polisemia) e viceversa (sinonimia). Esistono modelli più raffinati che in base ad operazioni di algebra lineare cercano di cogliere correlazioni tra vocaboli in modo da disaccoppiare maggiormente la semantica insita in un testo dal veicolo lessicale (cfr. paragrafo 3.2).

II.3.2 Stima della semantica associata a una collezione

La semantica associata ad una collezione D_i di documenti di testo può essere modellata con il vettore centroide $\vec{\mu}$, risultante dalla media dei vettori $\mathbf{x}$ associati a ciascun documento:

$$\vec{\mu} = \frac{1}{|D_i|} \sum_{\mathbf{x} \in D_i} \mathbf{x} \tag{3}$$

Le esigenze informative dell'utente possono essere stimate con il centroide $\vec{\mu}$ della collezione degli articoli scientifici D_i.

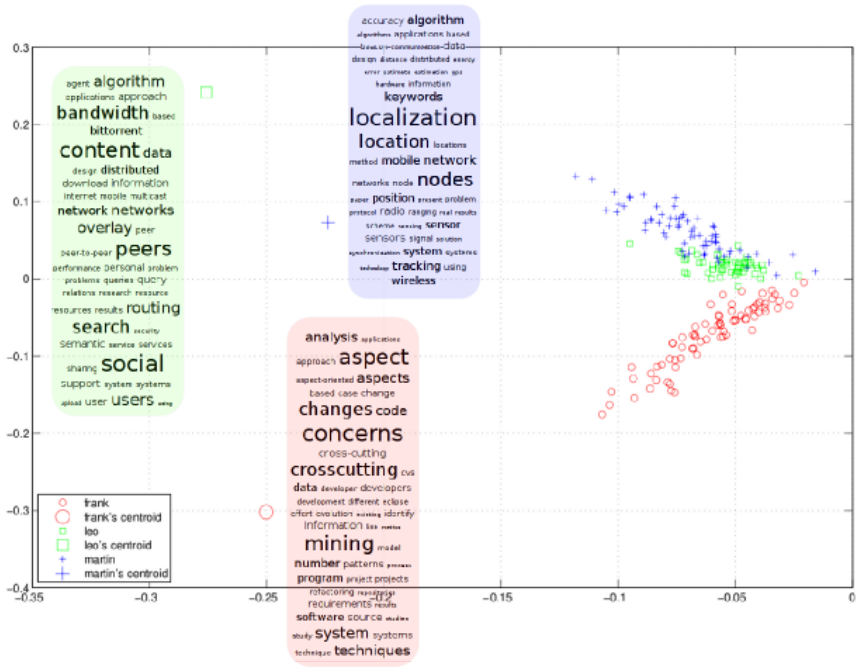


FIGURE 0.3: Proiezione in uno spazio semantico bidimensionale (cfr. paragrafo 3.2) dei vettori degli articoli scientifici relativi alle esigenze informative di tre utenti. Per ciascuna collezione è indicato il centroide dei documenti. Si noti che questo non corrisponde al contenuto semantico medio della collezione. Per ciascun centroide sono riportati i 50 termini di peso più elevato, secondo il modello statistico tf-idf.

II.4 Gestione del social network

Una popolazione di agenti, ciascuno caratterizzato da delle proprie esigenze informative, può essere organizzata in una topologia di interconnessione che rifletta la similarità tra le loro esigenze informative. Poiché queste sono supposte dinamiche, dipendentemente dalle loro esigenze individuali, la topologia di interconnessione è in continua evoluzione. La gestione di topologie in contesti dinamici e altamente popolati può essere effettuata in maniera completamente distribuita facendo affidamento a protocolli su base epidemica che garantiscono una buona efficienza a fronte di un basso livello di intrusione (cfr. sezione 4.4).

Le esigenze informative dell'utente sono sintetizzate nel profilo π dell'agente in base al quale il protocollo aggrega agenti con esigenze informative simili calcolate da un'apposita funzione di similarità ρ. Per motivi di efficienza nell'adattamento alle dinamicità della popolazione di agenti, si richiede che ρ sia una distanza definita su uno spazio metrico (cfr. paragrafo 4.4.2).

II.4.1 Profilo dell'agente

Il profilo dell'agente, che ne sintetizza le esigenze informative, dovrebbe essere di dimensioni contenute poiché da questo dipende il livello di intrusione del protocollo.

Nel precedente paragrafo II.3 abbiamo modellato le esigenze informative di un utente con il centroide $\vec{\mu}$ degli articoli scientifici letti. Questo è un vettore in uno spazio vettoriale ad alta dimensionalità, nel quale la base è data dal vocabolario. Al fine di contenere le dimensioni del profilo, il centroide è approssimato con un vettore a dimensione ridotta ottenuto selezionando le k componenti con peso w_{ij} maggiore (4a). La qualità dell'approssimazione può essere valutata dal rapporto tra le norme (4b):

$$\vec{v} = \{\mu_1, \mu_2, \ldots, \mu_k\},\ \mu_i \in (\vec{\mu}, \geq)_k \tag{4a}$$

$$\eta = \frac{\|\vec{v}\|}{\|\vec{\mu}\|} \tag{4b}$$

Il profilo $\vec{v}$ dell'agente così calcolato è quindi normalizzato e utilizzato dal protocollo epidemico di gestione della topologia al fine di individuare gli m agenti che abbiano le esigenze informative più simili, secondo la funzione di similarità ρ. Ciò richiede che i profili degli agenti siano vettori dello stesso spazio vettoriale, i.e. ciascun agente usi la stessa base. Ciò può essere aggirato accludendo nel profilo la base del vettore $\vec{v}$, i.e. il profilo dovrà contenere anche la lista dei k termini selezionati dal centroide $\vec{\mu}$.

Poiché la qualità del profilo influenza la topologia della rete di interconnessione, questo dovrebbe essere accuratamente valutato. Nella nostra emulazione abbiamo usato come profilo i 30 termini di peso maggiore selezionati dal centroide dei documenti.

II.4.2 Funzione di similarità

La similarità tra le esigenze informative degli agenti è calcolata come la distanza tra i corrispondenti vettori dei profili, in modo analogo alla similarità tra documenti secondo il modello a spazio vettoriale del testo.

$$sim(\vec{v}_j, \vec{v}_k) = \cos\theta_{j,k} = \frac{\vec{v}_j \cdot \vec{v}_k}{\|\vec{v}_j\|\|\vec{v}_k\|} = \frac{\sum_i v_{i,j} \cdot v_{i,k}}{\sqrt{\sum_i v_{i,j}^2}\sqrt{\sum_i v_{i,k}^2}} \tag{5}$$

Ciascun agente mantiene la lista degli m agenti con esigenze informative più simili. Le dimensioni di tale lista, che influenza la topologia della rete di interconnessione, dovrebbero essere accuratamente valutate al fine di ottenere proprietà desiderabili in termini di località degli interessi informativi.

II.5 Ricerca e raccomandazione di contenuti

Il servizio di ricerca e raccomandazione di articoli scientifici è realizzato con il supporto della rete di interconnessione tra gli agenti che riflette la prossimità tra le esigenze informative degli utenti.

Quando un utente interroga l'agente con una query che ne descrive le esigenze informative, l'agente ricerca contenuti rilevanti inoltrando la richiesta agli agenti vicini in accordo alla topologia di interconnessione. Tale modello di ricerca risulta efficiente qualora le richieste dell'utente avvengano secondo una logica di continuità che possa essere prevista dall'agente.

Anziché usare questo modello di recupero dell'informazione, l'agente può invece proporre in modo proattivo all'utente contenuti ritenuti rilevanti in accordo alle sue esigenze informative. Ciò è effettuato interrogando il vicinato con il profilo dell'utente. Questo schema consente di adattare i contenuti resi disponibili dalla popolazione di agenti alla specificità di ciascuno di essi.

Compendio III

Conclusioni e direzioni di sviluppo

Questo capitolo riassume le motivazioni alla base di questo lavoro e sintetizza le soluzioni che sono state proposte. Il paragrafo III.2 discute delle analogie con sistemi similari, mentre il paragrafo III.3 traccia le linee guida per lo sviluppo futuro del progetto.

III.1 Riepilogo e contributo

La localizzazione di contenuti è un problema di carattere generale nelle reti p2p. I più diffusi sistemi p2p di distribuzione di contenuti delegano il servizio di ricerca a un sistema basato sul modello client-server nel quale il server ha una visione globale dei contenuti disponibili. Ciò richiede in generale un considerevole investimento in termini di infrastrutture al fine di ottenere elevati livelli di scalabilità. Un tale approccio è inoltre incline al sorgere di problematiche legate alla sicurezza ed alla riservatezza dei dati che sono concentrati in un unico punto.

III.1.1 Contributo

Questo lavoro ha avuto come scopo l'indagine dei modelli alla base di un sistema informativo completamente decentralizzato che offra a utenti accademici un servizio di ricerca e di raccomandazioni di articoli scientifici. L'approccio utilizzato consiste nell'organizzare gli agenti partecipanti in una topologia di interconnessione che possa essere sfruttata per facilitare la localizzazione di contenuti messi a disposizione da ciascun agente.

Facendo uso di un protocollo epidemico a bassa intrusione la topologia di interconnessione tra gli agenti viene proattivamente e dinamicamente adattata alle loro esigenze informative. Tale topologia individua un social network che aggrega utenti con simili esigenze informative, le quali sono automaticamente stimate sulla base dell'analisi delle loro abitudini di lettura. Ciascun utente colleziona local-

mente gli articoli scientifici letti, dai quali vengono automaticamente estratte una serie di informazioni strutturate, quali il titolo, l'abstract, l'intero contenuto testuale e le citazioni, che vengono localmente indicizzate. Le esigenze informative dell'utente possono essere stimate analizzando il contenuto di questo database. Date le caratteristiche essenzialmente testuali degli articoli scientifici, la semantica associata può essere modellata facendo ricorso al modello a spazio vettoriale del testo che si basa sull'analisi statistica delle ricorrenze lessicali nel testo. La similarità semantica tra due documenti testuali, modellata su base lessicale, è riconducibile al computo di distanze tra vettori di uno spazio vettoriale. Le esigenze informative dell'utente vengono stimate computando semplicemente la media della semantica associata a ciascun documento. Tali esigenze informative, modellate da un vettore in uno spazio vettoriale ad alta dimensionalità, vengono approssimate con un vettore a dimensione ridotta ottenuto selezionando le k componenti di maggior peso. Ciò è necessario affinché il protocollo epidemico adottato per la gestione della topologia abbia una bassa intrusione. La ricerca e la raccomandazione agli utenti di contenuti ritenuti di interesse è basata sfruttando il social network che, per come è costruito, soddisfa localmente le esigenze informative degli utenti, dipendentemente dalla qualità della stima effettuata.

III.2 Discussione e critica

L'approccio alla soluzione adottata è ispirata al servizio distribuito di ricerca e raccomandazioni adottato dal sistema di file sharing Tribler [PGW+08], che aggrega utenti con esigenze informative simili, stimate sulla base dell'analisi della storia dei file scaricati. Il nostro sistema, che ha come oggetto informativo documenti essenzialmente testuali, differisce da questo per le modalità con le quali il sistema stima le esigenze informative dell'utente.

Mendeley[1] è un sistema di raccomandazioni per articoli scientifici, ancora in fase beta di sviluppo, con caratteristiche molto simili, per gli scopi che si propone, al nostro progetto [HR08]. Differentemente dalla nostra soluzione, che è completamente decentralizzata, Mendeley colleziona globalmente le preferenze di tutti gli utenti facendo riferimento al classico modello di computazione client-server.

III.3 Direzioni di sviluppo

In questo lavoro abbiamo considerato soluzioni che si sono limitate a tenere in considerazione gli aspetti fondanti di un sistema informativo completamente centralizzato per la ricerca e la raccomandazione di articoli scientifici.

Le direzioni di sviluppo dovrebbero perseguire lo stato dell'arte nella realizzazione di ciascun componente del sistema. In particolare, per ciò che riguarda

[1] https://www.mendeley.com

l'estrazione di informazioni strutturate da documenti orientati alla presentazione, quali i documenti in formato PDF, lo stato dell'arte prevede l'utilizzo di sistemi di apprendimento automatico. Un approccio analogo dovrebbe essere adottato anche dal sottosistema di apprendimento delle necessità informative dell'utente, che dovrebbe basarsi sull'analisi di una serie di misure oggettive derivanti sia dall'interazione dell'utente con il sistema (e.g. il tempo impiegato nella lettura dei singoli articoli, le parti sulle quali l'utente si è maggiormente soffermato, l'analisi delle query sottoposte al sistema, etc.) e sia dall'analisi degli articoli letti. Per quanto concerne quest'ultimo aspetto, dovrebbe essere preso in considerazione un modello più accurato della semantica del testo, quale l'analisi della semantica latente (*Latent Semantic Indexing*) che ovvia alle limitazioni del modello lessicale della semantica adottato in questo progetto.

Poiché la topologia della rete di interconnessione dipende dal profilo dell'utente e della metrica impiegati dal protocollo epidemico, questi dovrebbero essere opportunamente valutati al fine di ottenere topologie che supportino la scalabilità e la località dei contenuti, e.g. qualità del clustering.

III.3.1 Visione

Il modello di computazione p2p permette di ridurre i costi associati alla collaborazione tra gruppi di utenti. Questo potenziale potrebbe essere sfruttato in ambito accademico da una piattaforma basata su un social network che faciliti la collaborazione tra gruppi di ricerca con simili interessi attraverso la condivisione di idee ed esperienze.

Peer-to-Peer search and recommendations of scientific literature

Chapter 1

Introduction

Peer-to-peer (p2p) distributed computing model enables the design of very large applications with very low cost. Content location is a general problem in p2p networks. Recent solutions have proposed to organize network topology around data semantics and to exploit the resulting topology in order to facilitate content location. By resorting on this latter model we craft the foundations for a fully decentralized search and recommendation system tailored for scientific literature.

Section 1.1 introduces the peer-to-peer computing model and scope of usage. Section 1.2 discusses the problem of locating content in distributed systems and highlight some solution which have been proposed. Section 1.3 specifies system requirements and introduce our approach to the solution. Section 1.4 outlines the thesis content organization.

1.1 Peer-to-Peer computing model

Peer-to-Peer (P2P) computing model is a class of distributed architectures which are designed by interconnecting nodes with the purpose of sharing their resources such as content, computational power, storage, bandwidth or even human presence by direct exchange rather than requiring the intermediation of a centralized server. Communication between peers is completely symmetric meaning that nodes act as both clients and servers. This leads to the design of fully-decentralized service in which there is no central server coordinating the operation of the entire network.

P2P computation relies on a network of nodes connected between them. This network is formed on top of, and independently from, the underlying physical computer (typically IP) network and thus is referred as an *overlay network*. The topology structure, degree of centralization and routing and location mechanism it employs for messages are crucial to the operation of the system as they affects its fault-tolerance, self-maintainability, adaptability to failures, performance, scalability and security [ATS04].

1.1.1 Scope

Client-server model of computation partitions a task between service provider (server) and service requesters (clients). While scaling a service based on a client-server architecture is technically feasible (e.g. Google, YouTube, Wikipedia, Facebook) it requires a sizeable capital investment in the infrastructure. This approach can only be adopted when there is an underlying business model to the application. Moreover client are highly dependent on centralized entities with complete authority are therefore prone to become the victims of possible exploitation.

In contrast P2P computing model leads to decentralized designs which need no large infrastructure expenditures since the system is self-administered and does not depend upon any central entity, which means that the service is handled by the peers themselves. Administration maintenance, responsibility for the operation and even the notion of ownership of P2P system are distributed among the users instead of being handled by a single company.

Ability to function, scale, and self-organize to deal with fault and in presence of high dynamics, without the need of a central server and the overhead of its administration are key concerns while designing a P2P systems. Large scale P2P networks exhibit properties comparable to biological and social entities and a suitable approach consist on modeling them as societies resorting to models borrowed from social theory such computational sociology and evolutionary economics [MH08].

1.1.2 Content distribution systems

Content distribution systems (e.g. Napster, Gnutella, BitTorrent) have been the killer application for P2P computing model. These systems rely upon overlay network made up high dynamic node population in which resources, such as contents and bandwidth, should be donated by each peer. This leads to a complete decentralized service which can achieve the same performance of high-cost central solution architectures.

1.2 Content location

Content distribution presuppose that content in which a user is interested has been located since it is spread out over the network. This task is usually carried out by an information retrieval service which provides to a user the content which are expected to be relevant according to its needs. This service usually relies on a global view of the content from which it pick up those items which are expected to be relevant to the user according to a certain relevance model. Such a system, which requires a global view of the content, can be easily mapped into a client-server service in which the server has a global knowledge of contents.

Some popular P2P content distribution systems such as the former Napster and BitTorrent, tackle the concern of locating content by delegating this task to a client-server service.

The main difficult while resorting to P2P computational model lies in the cost required on providing a global view of the network to each peer. Solutions which rely on broadcasting are inherently not scalable [CRB+03]. Solutions which rely on distributed hashing middlewares have been used in order to map systems conceived as central services into fully-distributed computation. However this approach leads to high overhead cost which is not suitable while dealing with high dynamics large size networks such as content distribution P2P systems [RV03, LLH+03].

Recent solutions have proposed to organize network topology around data semantics and to exploit the resulting topology in order to facilitate contents location [RM06]. Since data and node population are supposed to have high dynamics, network topology should evolve consequently. Epidemic protocols have been used as the basis of lightweight and reliable services for managing topologies in these settings [VvS05]. Tribler is a P2P file sharing system built on top of the BitTorrent protocol which forms a semantic overlay network to identify related content, facilitate search, and recommend content to user without the need of a central service [PGW+08]. Network topology emerges around user's information needs which are automatically learned by tracking user's download preferences. Resulting topology is exploited in order to recommend contents and facilitate search.

Clustering together users with similar information needs leads to the construction of a social network in which friendships emerges around data. Such distributed social network approach may be considered as a framework on top of which many application which claims for personalization, collaborative filtering and ontological classification can be developed [AHPE09].

1.3 Contribution

Purpose of this research consists of crafting the foundations for a P2P search and recommendation system tailored for scientific literature. We will focus on solutions suitable for highly-populated highly-dynamics P2P networks such as those employed for content distribution systems.

Search and recommendation should be considered as a service which can be developed on top of a social network in which friendships are based on similarities in user's information needs. By identifying people with similar research interest a social network for researchers can be built. This can be exploited in order to facilitate search and for recommending interesting articles and researcher to the user [HPK08].

We resort to the same model adopted by Tribler file-sharing systems though

modifications application-tailored are required in order to capture the notion of researcher's information needs. The relevance model used for file-sharing system, which relies on download history, is not suitable for text-based documents since two users may found relevant the same subject even if they did not read the same papers. Semantics of text-based documents can be learned by relying on statistical model of language, such as the vector space model of text. By using this model the relationships between papers and users can be exploited for locating relevant content. Recommendations would thus help researchers to discover literature that could be of interest to them.

1.4 Thesis outline

This work is organized around the three key concerns which crop up while designing an information retrieval system: how data are represented, how the user expresses the query and how the system satisfy the user's information needs.

Chapter 2 concerns on finding a data model for scholarly literature and discusses about trade-offs while claiming for an automatic metadata extraction from presentation-oriented documents such as PDF files. We designed and implemented a prototype of a component which automatically extracts metadata, full-text and cited references from research papers in order to set up a local scientific literature database.

Chapter 3 reviews models and techniques which have been traditionally employed while analyzing text documents with links between them, such as scientific literature or Web pages. A retrieval model should rely on these techniques while modelling the notion of document relevance with respect to user's information needs.

Chapter 4 discusses about concerns on locating contents and points out problems while claiming for a fully decentralized approach. We review models which have been traditionally employed while distributing an information retrieval and we highlights their limits while dealing with highly-populated, highly-dynamic P2P networks. In order to meet our requirements we consider a solution which dynamically organizes network topology around data semantics by relying on a gossip-based protocol.

Chapter 5 discusses about the design of a proof-of-concept retrieval model tailored for scientific literature which can be used for learning the user's information needs. We specify message which should be employed by a gossip-based protocol in order to proactively build a social network around user's information needs. We implemented a prototype to prove that the proposed solution can be designed to meet our goals.

Chapter 6 gives an overview of the project's contributions and points out some ideas for future work.

Appendix A contains an analysis of the bibliography of this thesis which gives a briefly idea of its content. Analysis has been done by using the tools developed while designing the retrieval model.

Chapter 2

Data model and mining techniques

This chapter focuses on providing a data model for the data we are wishing to search for. Since data model granularity affects search capabilities, it is desirable to have a fine-grained model which captures as many features as can be observed.

In order to build a database we need to collect these features, i.e. structured data, from underlying data source. Since nowadays most of scholarly literature comes as unstructured text (e.g. pdf documents) we have to deal with the difficult of extracting structured data from those kind of documents.

In section 2.1 we discuss about these trade-offs and we suggest a data model for scholarly literature. Different solutions can be employed for collecting specified structured data from unstructured pdf documents. Section 2.2 discusses about means which relies on Web mining or collaborative solutions and highlights some key-concerns. Section 2.3 outlines the scenario of automatic extraction of structured data from unstructured text document. By pursuing this latter option we designed a prototype module which automatically extracts from pdf papers the data required by our defined model. In section 2.4 we discuss about design choices and in section 2.5 we assess its effectiveness.

2.1 Data models for scholarly literature

The definition of a proper data model for an information retrieval system is a key concern since it affects techniques and strategies which can be employed while designing the information retrieval subsystem. That subsystem is in charge to retrieve the best items out of the whole collection which fit the user's information needs.

A fine-grained model, which takes into account more aspects of the data that we are wishing to search for, offers much rich information to the information retrieval model. However, because we assume that the collection we are going to search within is made up essentially of unstructured text document such as

classic PDF files, providing a rich model from this underlying source objects can be difficult to obtain.

2.1.1 Design requirements trade-offs

Nowadays scientific research data and results are shared primarily through publishing paper. Researchers encapsulates concept and ideas within a paper by coding them in natural language which is tailored for human consumption. Our design choices will be driven by trade-offs to offer to the information retrieval model a rich data model which can be efficiently exploited and the feasibility to obtain that data model.

A simple solution may be to model a scientific work as a simple bag of words. In such hypothesis search capabilities will be limited by this strong assumption. Typically they rely on using just the Vector Space Model (cf. 3.1). At the opposite side we can consider a much more fine-grained model in which we model a scientific publication as an ontology, i.e. a formal representation of a set of concepts within a domain and the relationships between those concepts. Although this solution should yield in a rich set of possibilities while searching, adopting that model for unstructured text documents can be very difficult. The common adopted solution relies on artificial intelligence techniques which aims to deal with natural language.

Though the semantic web has been proposed as a general solution to this [BLH01], it is currently a largely unrealized vision of the future. The new semantic web technology may change the way scientific knowledge is produced and shared. Moving towards this direction will facilitate data sharing, discovery and integration. The challenge is how to get the scientific community to publish experiment data on the Web in an appropriate format.

2.1.2 A proposed model

Our design choices are driven by aforementioned constraints which calls for a trade-off between search capabilities and feasibility of automatically build a database from underlying source of unstructured text documents such as PDF files.

Bibliographic data Each scholarly work can be characterized by a title and one or more authors which wrote it. Usually its content is summarized in a short section, i.e. the *abstract*. The document body contains the explanations of the concepts and ideas.

Citations Each scholarly work has usually a section, called *bibliography*, which lists reference to previous works which are strongly related to it. In order to identify with a reasonable accuracy a previous published work, authors usually

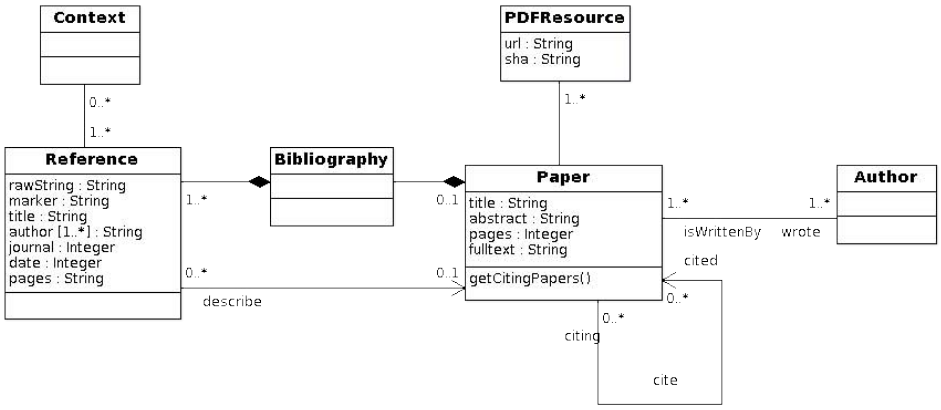


FIGURE 2.1: Class diagram of a collection of papers. Each paper has its bibliographic data such as the title and the authors. We gives back semantics to customary text regions such as the abstract and the body. The bibliography is a set of citations strings, each of them is made up of a number of pre-defined field which aims to describe the referred paper.

provides a description in natural language of the work, i.e. the *citation string*. By using that description a reader could try to retrieve the described resource. Note that if it exists a way to uniquely and permanently identify each scholarly work, each author could simply provide that identifier rather than a natural language description of the referred work.

Use of citations have been widely discussed in order to analyze scholarly literature [Gar79, Gar67]. They encapsulates authors' knowledge of literature and they should be exploited by an information retrieval model.

Citations between papers are similar to links between HTML Web pages. They allow to treat a collection of linked object as a directed graph for which graph theory can be used to exploit encoded information.

2.2 Mining data from unstructured text document

The adoption of the model depicted in figure 2.1 depends on the capability of a machine to collect, with the minimum human effort, the required structured data from underlying source of unstructured text documents in order to build a searchable database. Two different approaches can be taken.

Retrieving metadata from the Web The first option resorts to the usage of the Web in order to mine the necessary data to link with a given scholarly work. This approach works if the information which we are looking for is already available

in some structured form. It makes no advantage to gather required data from unstructured data available from the web. As example many digital libraries expose their meta-data according to the Open Archive Initiative interoperability standard [LVdS01]. Another solution could be the one that resorts to some forms of collaborative human efforts to collect the meta-data such as the solution proposed by the Web-based social bookmarking tool CiteULike[1].

The adoption of this solution requires to address two key concerns. Section 2.2.2 discusses about problems of finding a common representation of meta-data across many sources in the Web, while section 2.2.3 discusses about problems on uniquely identify documents across many users on the Web.

Mining metadata from unstructured source The latter option aims to mine the required structured data by analyzing the unstructured text document source. Since this requires to deal with natural language, the common adopted approach relies on artificial intelligence techniques. We discuss this option in the next section 2.3.

2.2.1 Unstructured text documents

Nowadays most of the scientific literature available over the Web is in the PDF or in HTML formats, which were originally designed mainly for human consumption. A machine can easily display their content, but it can not easily capture the knowledge coded within them since is is better at handling carefully structured and well-designed data.

We will focus our analysis on PDF format because it has the most spread. Nevertheless is more difficult to mine the data from PDFs rather than from HTML since PDF format is much more oriented to the document presentation while HTML describes to a certain extent also the semantics of a document by using structural mark-ups (e.g. `<h1>Title</h1>`). Structural markups does not denote any specific rendering altough most Web browsers have standardized on how elements should be formatted.

Portable Document Format (PDF) Pdf is a file format originally developed by Adobe Systems for document exchange which has became an open standard[2]. It is used for representing two-dimensional documents in a manner independent of the application software, hardware, and operating system. Each pdf file encapsulates a complete description of a fixed-layout 2-D document that includes the text, fonts, images, and even 2-D vector graphics which comprise the documents. The openness and fidelity to paper of this standard has lead the pdf to become the ubiquitous format for electronic publishing of academic papers.

[1] www.citeulike.org

[2] It has been published on July 1, 2008 by the ISO as ISO 32000-1:2008

Embedding metadata A straightforward solution to retrieve structured data from unstructured text documents such as PDF is to couple within it some extra information. Although this option has the inherent limitation of requiring an extra effort on coupling data, which could be done by authors or collectively updated by the user, it has the advantages of keep together the digital resource with its machine readable metadata which avoids problems of identifying resources as we discuss in section 2.2.3. However this option requires a standard format for storing metadata.

2.2.2 Metadata representation

The PDF standard has the ability to store a pre-defined and limited set of metadata in the Document Information Dictionary (DID). It has fields like *title*, *authors*, etc. Perhaps because of these strictly limitations, these fields are extremely rarely used in the academic world.

Adobe XMP Adobe has produced a relatively new open standard for storing metadata which can be used for PDFs and for a wide variety of file formats as well. The eXtensible Metadata Platform (XMP) defines a metadata model which can be used with any defined set of metadata items. The most common metadata tags recorded in XMP data are those from the Dublin Core Metadata Initiative[3], which includes things like title, description, creator, and so on. The standard is designed to be extensible, allowing users to add their own custom types of metadata into the XMP data.

Reference manager software metadata formats Currently many scholars and authors manage their knowledge of the scientific literature by using some kind of computerized reference management system such as BibTEX, EndNote, Reference Manager, RefWorks. These software are designed mainly for recording and utilising bibliographic citations. Each scholarly work is described by a set of bibliographic items. For instance, the popular BibTEXtool uses some standard data entries depending on the kind of the scholarly work, such as articles, books, theses. As example the BibTEXcode snippet depicted in figure 2.2 describes an article published by a journal. Besides the standard data entries other valuable information can be stored by defining customized data entries. As instance we can store the abstract section by using the key `abstract`. By following that idea we could manage almost the same information retained by the model depicted in figure 2.1.

Lack of standardization Instead of using the BibTEXor EndNote file formats, we could retain the same information by defining an XML schema and storing the

[3]RFC2413: Dublin Core Metadata for Resource Discovery

```
@article{bernerslee2001psw,
  title={{Publishing on the semantic web}},
  author={Berners-Lee, T. and Hendler, J.},
  journal={Nature},
  volume={410},
  pages={1023--1023},
  month={April},
  year={2001}
}
```

Figure 2.2: Snippet example of the content of a BibTeX `.bib` file that stores the bibliographic information of a paper.

data according to that definition, e.g. by relying on Adobe XMP. However BibTeX, and EndNote formats are used by wide communities, although none of them has been recognized as universal standard. Also some popular digital libraries such as Google Scholar, IEEE XPlore and ACM digital library expose the metadata of their indexed documents by using these formats.

2.2.3 Uniform Resource Names

Even if we have identified an established format for describing metadata, if we keep them as a separate object we need a mechanism for linking the metadata with the described document. Basically we need to identify each document by using a Uniform Resource Name (URN) that is intended to serve as persistent, location-independent resource identifier[4]. URNs are required to remain globally unique and persistent even when the resource ceases to exist or becomes no longer available.

Standard URN proposals Nowadays there is no single way of identifying publications across all digital libraries on the Web. Although various identification schemes such as the Digital Object Identifier[5] (DOI), International Standard Book Number (ISBN), PubMed identifier (which is used by PubMed digital library) and many others exist, there is not yet one identity system to rule them all. A DOI is a URN that allows persistent and unique identification of a publication, independently of its location. Despite their usefulness, DOIs has not yet been wide adopted.

[4] RFC 3986 Uniform Resource Identifier (URI): Generic Syntax
[5] http://www.doi.org/

Non-standard solutions We could also consider the use of non-standard URNs, i.e. identifiers that don't use officially registered name-spaces[6]. As instance we can use the digest of the document, which can be computed by using an hash function, as its URN. With this solution we are confusing the abstract idea of the resource, with its distribution media. For example we can encode a given scholarly work by using a different version of files which have different digests. It is not difficult to find on the Web different files which encodes the same paper.

Another interesting solution is the one adopted by [Ber00] in which URNs for scholarly works are built by using its bibliographic informations such the title, the first author, the journal and the year of publication. Of course the viability of this option depends on the availability of high quality metadata. In section 2.4.3 we discuss some key advantages of using this solution while building the citation graph.

2.3 Automatic information extraction

Human-readable text documents such as PDF usually provides only a visual description of their content so the logical structure of the document is encoded by authors relying on typographical conventions. Different techniques can be used in order to give back a structure to the document. Information extraction is a type of information retrieval whose goal is to automatically extract structured information, which types has been pre-specified, from unstructured or loosely structured machine-readable documents.

Section 2.3.1 gives a formal description of the information extraction task as a chunk classification problem and points out common adopted solution. Section 2.3.2 briefly reviews machine learning approaches and indicates previous works which have made use of them.

2.3.1 Problem statement

We consider the text content within a document D as a set of text chunks t_i (i.e. *tokens*) with no associated semantics. Given a pre-defined set of *data items* $\{k_1, \ldots, k_N\}$ some tokens t_i within D can be labeled according to those data items. For each pre-specified data type k_i we can define an equivalence relation $\sim$ on D which induces a partition on the set D into two subsets which can be labeled as *relevant* and *non relevant* with respect to the equivalence relation $\sim$.

$$D_j := \{t_i \in D | t_i \sim k_j\} \tag{2.1}$$

The feasibility of an automated information extraction system relies on the definition of those equivalence relations, one for each of the pre-specified data types.

[6] according to RFC 2141 (URN Syntax) and RFC 3406 (URN Namespace Definition Mechanisms)

Problem definition This task belongs to a class of problems that are difficult to solve using traditional algorithms. This is typically caused by one or more of the following factors:

- difficult in formalizing the problem. For example a human can be easily recognize and classify the different sections within a paper, but it is hard to describe a sequence of computational steps that performed on the text and its visual properties allows a computer to do the same work;
- a high number of variables. As instance, there is no a unique standardized way to locate the various sections within a paper. Usual methods to distinguish different sections rely on different fonts properties, such as size, font face and so on, use of spacing or different page alignments;
- lack of theory;
- the need for customization. For example, if we are going to classify documents as whether relevant or not, the distinction may depends significantly by the specific user.

Common adopted solutions Several approaches have been proposed for automatic metadata extraction, with the most common tools including regular expressions, rule-based parsers, and machine learning algorithms. Regular expressions and rule-based parsers could be easily implemented and can perform acceptably well if data are well-behaved. They rely on the use of heuristic algorithms which are a class of algorithms that are able to produce an acceptable solution to a problem in many practical scenarios, but for which there is no formal proof of their correctness.

2.3.2 Machine learning algorithms

Machine learning algorithms are used to automatically induce models, such as rules and patterns, by using observations (experiences, data, patterns). By receiving feedback on the performance, the learning algorithm adapts the performance element to enhance its capabilities. Depending on the feedback we can distinguish between the following forms of learning:

- Supervised learning. The learning algorithms receives inputs and the correct outputs, and searches for a function which approximates the unknown target function.
- Unsupervised learning. The agent receives only input data and uses an objective function (such as a distance function) to extract clusters in the input data or particular features which are useful for describing the data.

- REINFORCEMENT LEARNING. The agent receives an input and an evaluation (reward) of the action selected by the agent, and the learning algorithm has to learn a policy which maps inputs to actions resulting in the best performance.

Common adopted solutions Information extraction has been used in many applications such as information gathering in a variety of domains, automatic annotation of web pages for semantic web, knowledge management. Digital libraries such as CiteSeer [LGB99, GBL98, BLG98] and Google Scholar automatically mine metadata from crawled PDF documents without human intervention.

Machine learning methods used for information extraction tasks [Die02] include inductive logic programming, grammar induction, symbolic learning, Hidden Markov Models, Conditional Random Fields [PM04] and Support Vector Machines (SVMs). SVMs are becoming increasingly popular tools for classification as they have been proven to be effective for chunk identification and named entity extraction [HGM+03, KM00, SMR99].

Evaluation An ideal information extraction system should use the equivalence relations stated in equation 2.1 for automatically classification of the data chunks. Because the afore-mentioned reasons each of the methods that have been used yields to an approximation $\sim_a$ of the ideal equivalence relations. We will discuss how to evaluate the extent of this approximation, and therefore the effectiveness of the automated information extraction task, in section 2.5.

2.4 Designing an information extraction system for PDF scientific literature

We assume that a user has a collection of scholarly works stored as unstructured text documents such as PDF format. In order to make the collection easy searchable we need to build a database by adopting a model of the data such the one proposed in section 2.1.2 and depicted in figure 2.1. Structured data can be collected resorting on Web repositories, collaborative solutions or by mining the required data directly from the source documents. Hybrid solutions could be employed as well, in which data are mined mainly by relying on automatic information extraction and resorting to alternative solution, such as collaborative methods, in case it fails by missing some data.

We designed a prototype of a system which automatically builds a database, which uses the defined data model, by mining required structured data by analyzing PDF source documents collected by the user.

Designing a state-of-the-art module in charge of performing this task is out of the scope of this work, which aims mainly to provide a proof-of-concepts of the

distributed content search and recommendations of text data. This task is intended to be as a fundamental support to every information retrieval system apart to distribution issues. Therefore our design choices are driven by fast-prototyping constraints and some adopted solutions does not perform the state-of-the-art and should be investigated further.

Overview Section 2.4.1 discusses about concerns of accessing data from visual layout documents such as PDF. In order to ease the analysis we choose to convert PDF document to HTML. This choice will be the bottleneck which limits the effectiveness of the information extraction.

Section 2.4.2 discusses about means of extracting bibliographic information by relying mainly on text typesets. We design a simple rule-based parser in which we manually defined the rules by using heuristics. Although this solution performs quite good for well-behaved data, it is not easy adaptable to new data. A state-of-the-art solution should make usage of learning-based methods.

While segmenting each reference strings into sub-fields we employed the freely available ParsCit[CGK08] package which relies on Conditional Random Fields which are the state-of-the-art while dealing with text chunks categorization.

Section 2.4.3 discusses about meaning of automatically building a directed graph over the collection by using citations. We employed a filter&refine algorithm which resorts on a keyword search algorithm for filling in the filter stage and on some heuristics for the refine stage.

An evaluation of the effectiveness achieved by our prototype is discussed in section 2.5.

2.4.1 Parsing PDFs

A document page encoded in PDF format is represented by a collection of primitive objects, which can be characters, simple graphics shapes, or embedded objects. Each primitive has properties, such as font size and type for characters, and position on the page, given as coordinates of the object's bounding rectangle. For example given a document, we could see that the text content is mixed with PDF's instructions. Moreover spaces are not explicit, but they are coded implicitly in terms of the placement of words on the page. Whole words are not always bracketed together: to give greater control over spacing, letters and words fragments are often placed individually. In order to mine and to categorize data from PDF documents we can take one of the two following approaches.

Direct parsing

The first option is to direct parse and analyze the information retained by PDF. We can have access to each element stored in the page and its corresponding

properties. For instance, `PDFMiner` is a PDF parser and interpreter written entirely in Python that aims to help analyzing text data from PDF documents. It allows to obtain the exact location of text in a page, as well as other layout information such as font size or font name, which could be useful for analyzing the document. The task of information extraction should use all of these properties in order to identify and to categorize the various text chunks.

However, if we pursue this option we must also be concerned to group together sentences and paragraphs. Text extraction is complicated as PDF files are internally built on page drawing primitives, meaning the boundaries between words and paragraphs often must be inferred based on their position on the page. For these reasons, text cannot be extracted reliably by syntactic analysis of the PDF file.

Conversion to another format

The second option is to transform the original PDF document into a format more tractable to analysis. The two most common target formats are plain text and HTML. In general the conversion process is prone to errors and it could cause a loss of some valuable informations about the original document structure.

We tested many conversion tools in order to find the most suitable one. The tool `pdftotext`, which is part of the `Xpdf`[7] project, is an open source command-line utility for converting PDF files to plain text files. Because the simplicity of the target format all of the extra information, such as font size is lost. Mining and categorizing data from plain text without any kind of extra information (e.g. typesettings) can be very hard. First versions on CiteSeer [GBL98] used a modified version of `pdftotext` which inserts font tags into the output.

The tool `pdftohtml`, that is also based on the `Xpdf` package, converts PDF documents into HTML. It aims to build an HTML document that resembles the original document layout hence a huge amount of information is enclosed within the output document. Because it is easier to parse the HTML output than directly dealing with PDF, we chose this latter option. Whereas well-formed text extraction is difficult to do with PDF files, we exploit the well-formedness constraints of HTML format to ease the extraction process. We make use a standard SGML parser to gather the necessary data to be feed to the information classification task.

2.4.2 Extraction of bibliographic details

As described in section 2.3.2 the best approach for dealing with text classification relies on the use of machine learning algorithms, such as SVMs. However, because our fast prototyping constraints, we choose to implement this stage by using a

[7] http://www.foolabs.com

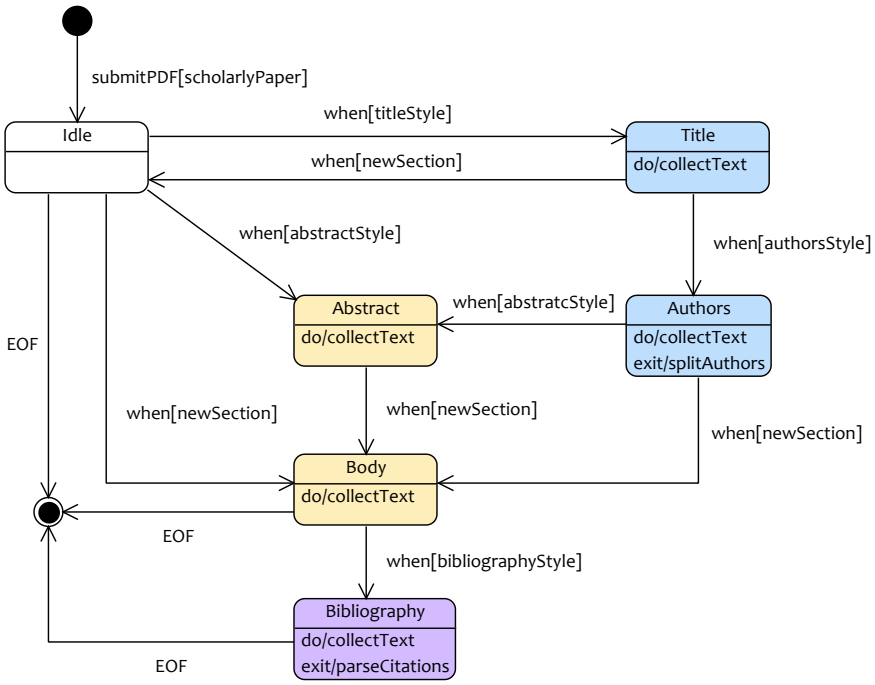


Figure 2.3: State diagram of the Finite State Automaton (FSA) employed for the categorization of text chunks which are enclosed within a paper. Each state encodes a well-defined data item. Transition conditions between states are implemented by using heuristics which aims to disclose the features that help in distinguishing the different data types. They rely on text typesets and regular expression in order to capture well-known text patterns.

simple Deterministic Finite State Automaton (DFSA) that relies on heuristics for carrying out the transition conditions between its states.

As described in the state chart diagram depicted in figure 2.3, we consider the document to be composed as a sequence of predefined usually recurring blocks: the header material, which comprises the title and the authors, the body, which can be split into sections, including the abstract that is the most important and recurring one, and finally the bibliography, which contains the reference strings. We encode the semantics of each block by using a different state of the FSA. Once a PDF document is submitted, we first convert it into HTML format for ease the access to the information, then we check for the existence of some of the specified text blocks in order to identify the submitted document as a scholarly work. If we recognize the document as a scholarly work we start analyzing the text chunks stream in order to classify them. The transition conditions between states are modeled by using an heuristic algorithm that relies on the use of some scoring functions which aims to estimate the likelihood that a given text chunk should

be classified as one of the predefined data types. For instance, referring to the state chart in the figure 2.3, if we are currently in the `Idle` state, we compute three different scoring functions, one for each transition towards not accepting state. We enable one of the available transitions according to the exceeding of certain thresholds by the scoring functions. We implemented the scoring functions using heuristic algorithms that rely on the use of regular expressions and text typesets such as font size, font face and position of the text on the page. Each scoring function is custom-tailored to the kind of data type, i.e. one of the FSA state, that we are going to recognize.

Exploiting text typesets Text typesets are an important features used by papers' authors to encode the document structure. We should leverage layout features in order to rediscover the document's logical structure. Since each journal has its own formatting style preferences, there is not a general rule to rediscover the document structure.

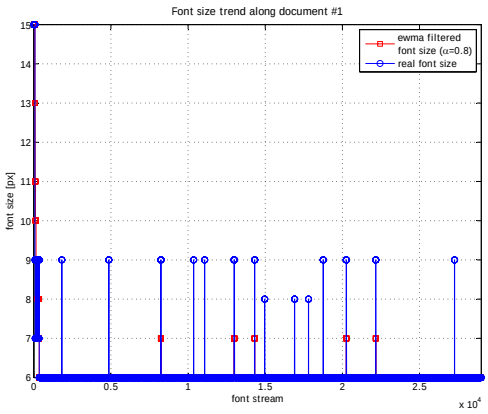
Authors usually rely on different font sizes to highlight the importance of text chunks. As instance, title and section titles are usually drawn in a rather larger font than the normal text. However this is not a general rule and a lot of variants are possible. As instance different sections could be also divided by using some extra vertical space or even different font faces and styles.

In figure 2.4 we analyzed two typical scenarios in which authors rely on different text typesets in order to encode the document structure. By treating documents as streams of typefaces, we plotted the text typographical features over all the typefaces in the document, from the first one, that usually belongs to the title, to the last one, which usually is part of the bibliography. We considered font size and the vertical placement on the page, computed from its top edge.

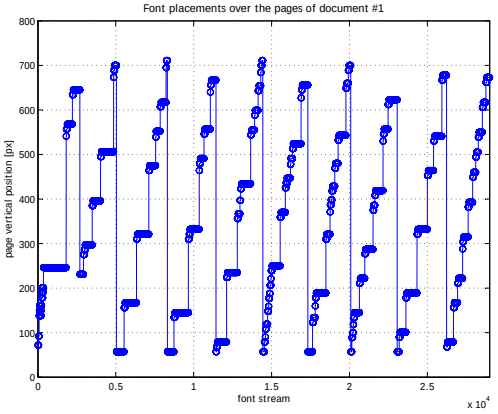
First document we analyzed makes large use of font size to give structure to the document. As we can see in figure 2.4(a) it is quite easy to distinguish the various sections and their relatively importance by using text font size properties. We can also distinguish the title, which is stood for the largest fonts, which are represented by the beginning markers, and the entire authors sections, which comprises the authors' names and their affiliation. In figure 2.4(b) we can distinguish the columns enclosed in the document. The last column comprises the bibliography section, which could be located by relying on the font size features. To a certain extent we can get a glimpse of the single citation strings.

Second document uses a different style to separate the various sections. As we can see in figure 2.4(c), it makes little use of font size. We can distinguish only the abstract and the reference sections, that are both entirely drawn with a smaller font size. Within the body is not possible to distinguish the various sections since the section title has the same size of the normal text[8]. As the previous case we

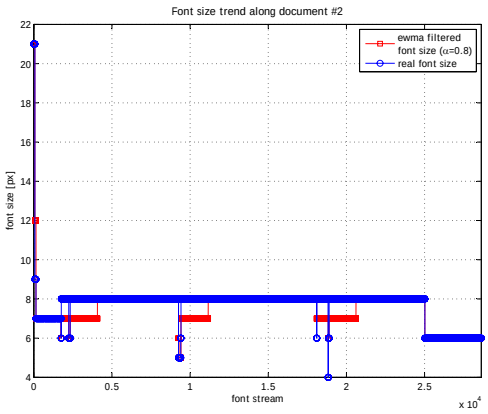
[8]The smaller fonts within the body section belong to figure's captions



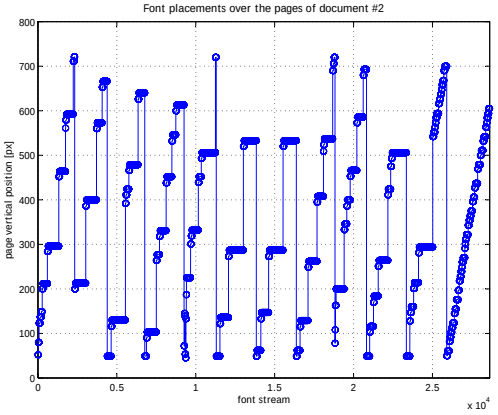
(a) Usage of font size through the document 1. The largest font size is used only for the title. Also the title sections and their relative importance are high correlated with the font size.



(b) Absolute vertical placement of characters through document 1. In the last column, which contains the bibliography, we can distiguish to a certain extent the single reference strings.



(c) Usage of font size through the document 2. Most of the sections sections within the document can not be located relying on font size. We can easily distinguish the title, the figure's captions and the references at the end that are drawn by using a smaller font.



(d) Absolute vertical placement of characters through document 2. Columns can be easily identified. Geometric positions can be leveraged for cluster words in the same sections. In the last two columns single reference strings can be identified.

Figure 2.4: Exploitation of text typesets for re-discovering the document's logical structure. We treat a document as an array of characters, each of which with associated typographical properties. We plotted some of these properties through all the document length (horizontal axis). We analyzed two different documents that portray two different scenarios on using text typesets for encoding the logical structure of the document.

can make out the single reference sections by observing the font placements in figure 2.4(d).

We are going to use these insights in order to implement the scoring functions for each of the predefined data types. However, because these wide range, it is not easy to find a general model which performs well with all the conceivable formatting styles. Machine learning algorithms approaches try to overcome the difficulty to find a model by automatically induce it by receiving feedback on the performance.

Title and authors The title of the document is usually written in the largest font size in the document at it appears at the beginning of the document. Although can be quite easy to identify the header material, which comprises besides the title also the authors and their affiliation, categorizing a text chunks as author can be quite hard. Because we did not find considerable distinguishing criteria to label the authors, we assume that they come next to the title. Therefore finding the authors section is dependent on the success of finding the title.

Abstract and body The keywords *Abstract* or *Summary* drawn in a rather large font than the document body is assumed to be a marker for the beginning of the abstract section. We also take into account the location within the document, that is assumed to be next to the front matter. The beginning of the body is assumed to be the first title section that the parser encounters after the abstract section. If the system can not reliable locate it, we assume the length of the abstract up to 1000 words.

Bibliography Most commonly references are found in a late section of an article. They are usually introduced by strings such as *References*, *Bibliography* or *List of References* or their common variations, which are usually drawn in a rather large font than the text content.

Once the complete reference section is extracted, the next phase is to segment individual reference strings. While there are various different styles for delineating individual citations (six major styles are detailed in [DTS+07]) we considered only the two most frequently used styles:

- citation strings marked with square bracket or parenthetical reference indicators.
- citation strings marked with naked numbers (e.g. "*1*" or "*1.*")

The first step is therefore to find the marker type for the citation strings. This is done by constructing a number of regular expressions matching common marker styles for the considered cases, then counting the number of matches to each expression in the reference string text. If either cases yields more matches than a

Style	Reference style example
IEEE	[1] S. Lawrence, C. L. Giles, and K. Bollacker, "Digital libraries and autonomous citation indexing," Computer, vol. 32, no. 6, pp. 67-71, 1999.
ACM	1. Lawrence, S., Giles, L. and Bollacker, K. 1999. Digital libraries and autonomous citation indexing. Computer, 32 (6). 67-71.
MISQ	Lawrence, S., Giles, L., and Bollacker, K. "Digital libraries and autonomous citation indexing," Computer (39:2) 1999, pp. 67-71.
JMIS	1. Lawrence, S.; Giles, L.; and Bollacker, K. Digital libraries and autonomous citation indexing. Computer, 32, 6 (1999), 67-71.
ISR	Steve Lawrence, C. Lee Giles and Kurt Bollacker, "Digital libraries and autonomous citation indexing," Computer, 39, 2, (1999), 67-71.
APA	Lawrence, S., Giles, L., and Bollacker, K. (1999). Digital libraries and autonomous citation indexing. *Computer*, *32*(6), 67-71.

Table 2.1: Examples of different journal reference styles (adapted from [DTS+07])

certain threshold the case with the greatest number of matches is indicated. In both cases, the same regular expressions that were used to find the marker type may be used to indicate the starting point of a citation, and citations are segmented in this manner. If no reference string markers are found, several heuristics, which make usage of relative position on the page and ending punctuation, are used to decide where individual citation strings start and end.

Parsing the reference strings

Each reference string is made-up as a set of fields (e.g., author, title, year, journal) that are represented as a string, with implicit cues such as punctuation to assist in recovering the encoded data. Although it is often straightforward for human readers to divide a citation into its constituent fields, the different grammars used to produce citations by different communities, coupled with inadvertent errors on the part of authors, makes this process difficult to automate.

Problem statement We first formally define the problem to be solved. We say that a reference string R has first to be broken down into a sequence of tokens $\{r_1, r_2, \ldots, r_n\}$. Each token has to be assigned the correct label from a set of classes $C := \{c_1, c_2, \ldots, c_m\}$. Evidence used in classifying some token r_i can be any data that can derived from the surface reference string, as well as previously-assigned $r_1, \ldots, r_{i-1}$ classifications.

Approaches to the solution This sequence labeling problem is common to a large of Natural Language Processing (NLP) tasks. Numerous studies have been

proposed in recent years. Those approaches can be classified into three categories: rule-based, knowledge-based and learning-based approaches.

Rule-based methods are widely used in real-world applications. CiteSeer [LGB99] uses heuristics to extract subfields. It identifies titles and author names in citations with roughly 80% accuracy [GBL98]. As a major drawback of the rule-based approach should be recorded that it requires a domain expert who has to explicitly formulate his knowledge about each reference style that should be recognized by the information extraction component. Therefore each style has to be manually analyzed first and then implemented in the system as a rule, dependent on the symbols already parsed, which is error-prone and time-consuming.

Knowledge-based methods work by using several template databases with various styles of citation templates. INFOMAP [DTS+07] is a hierarchical template-based reference meta-data extraction method with an overall average accuracy level of 92.39% for the six major citation styles. FLUX-CiM [CdSG+07] estimates the probability that a token can be labeled as a certain type by using the information encoded on an automatically constructed knowledge-base (KB). As example for identifying author names KB methods relies on an author name database so the quality of the database greatly affects the parsing accuracy. BibPro [CYKH08] does not need KB, only the order of punctuation marks in a citation string is used to represent its format.

Learning-based methods utilizes machine learning techniques (e.g. Hidden Markov Model (HMM) [Het08], Support Vector Machines (SVMs), Conditional Random Fields (CRFs) [PM04, CGK08], Probabilistic Finite State Transducers (PFST) [KKKB07]).

ParsCit ParsCit [CGK08] employs state-of-the-art machine learning models to achieve an overall word accuracy of 97.4% in reference string segmentation. It relies on the use of a trained Conditional Random Field model for label the token sequence within the reference string. CRFs are a probabilistic framework for labeling and segmenting structured data, such as sequences, trees and lattices. It relies on an undirected graphical model in which each vertex represents a random variable whose distribution is to be inferred, and each edge represents a dependency between two random variables. A common special-case graph structure is a linear chain, which corresponds to a finite state machine, and is suitable for sequence labeling. The distribution of each discrete random variable Y in the graph is conditioned on an input sequence X. The underlying idea is that of defining a conditional probability distribution over label sequences given a particular observation sequence [LMP01].

ParsCit classifies tokens within a set of of 13 classes, corresponding to common fields used in bibliographic reference management software (e.g. EndNote, BibTeX). It has been successfully deployed within CiteSeer$^{\chi}$ [LCB+06] and its source

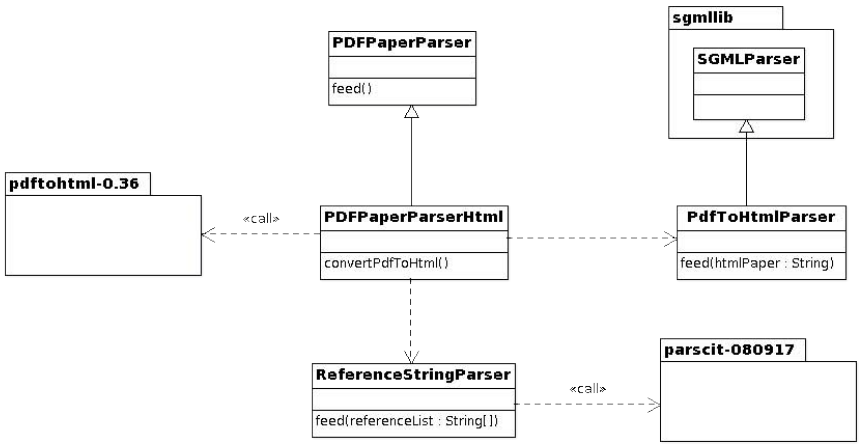


Figure 2.5: Class diagram of the PDF information extraction subsystem prototype.

code, written in Perl, is freely available to the community[9]. It rely on CRF++[10], an open source implementation written in C++ of Conditional Random Field.

We make extensive use of these packages for parsing reference strings and build the model depicted in figure 2.1. Nevertheless we have not yet considered performance issues. Because we chose Python as prototyping language, as a result the process of parsing citations is quite slow because we call a Perl interpreter for each citation that has been found in the document.

2.4.3 Building the citation graph

In the previous section we analyzed techniques for information extraction from visual layout documents such as PDFs in order to build a model for scientific literature. Hence we assume that once a PDF document is submitted, the system automatically gather the necessary information in order to build the model depicted in figure 2.1. We are going to use the information retained by the reference strings enclosed in the reference section in order to build a citation network. Each *reference string* is a comprehensive description of the cited work provided by the author in natural language, that helps on identifying the cited work.

Problem statement In order to treat a citation index as a graph [Gar67] we have a collection of scholarly documents $D := \{d_1, d_2, \ldots, d_n\}$ and a function T that we call *citing function*, mapping D into D, $T : D \mapsto D$. The expression Td_i will mean the set of papers which have cited the paper d_i. The entire graph is denoted

[9] http://wing.comp.nus.edu.sg/parsCit/
[10] http://crfpp.sourceforge.net/

by $G := (D, T)$. This states that the graph is composed of papers and a function which relates these papers.

Related works The automated linking of related information within a collection of document was first introduced by CiteSeer project [BLG98, GBL98, LGB99, LBG99] carried out at NEC research institute. Later, OpCit project [Ber00, BPZ01] and some other researches (e.g. [CLMR01]) addressed the problem. These were later followed by large scale academic domain citation system such as the well-known Google Scholar.

The viability and accuracy of an Autonomous Citation Indexing (ACI) system depends mainly on its ability to build a model, such as the one depicted in figure 2.1, from underlying source of documents which can be leveraged for building the citation graph.

Key issues Shaping the citing function T depends strictly on how authors provide citations to previous works and how scholarly works are being identified. There is an important difference between analyzed items and the works cited by those items. Both are writings, but while we know about analyzed items, references are to works which may or not be available.

Nowadays authors usually use reference strings for providing citations to previous work. Each reference string is a description in natural language that aims to identify a unique previous scholarly work by describing its bibliographical features such as the author names, the title, the journal in which it appears and so on. If there is no correlation between those bibliographical features and the URN that is used for identifying each work, we need to perform a search within a database of known works in order to identify if there is one that corresponds to the description provided by the reference string. In this scenario the citing function T should accept as input a reference string, split by its component fields, and should provide an identifier, if it exists, of the described work. The task of identifying the cited work by using a description of it, often inaccurate, in natural language, is a typical problem of information retrieval.

A different scenario is adopted by [Ber00]. It uses a non standard URN scheme for identifying documents which is built by using its bibliographic details. It builds each URN by concatenating three strings: the first author's last name, the 4-digit year of publication and the first 20 characters of the lower-cased title. This become a sufficient precise hash key for identify the cited work just by parsing the reference string. Despite its certain advantage in shaping the citing function T, its viability depends strictly on the availability of high quality meta-data.

Sometimes, especially in recent papers, authors include within each reference string a field that gives also the DOI reference to the cited work. If we identify each work by using a DOI (likewise any kind of URN), the resolution of the links by the citing function T becomes straightforward since it does not need to resolve

the name of the citing work relying on its description. However DOI URNs has not been yet wide adopted. Moreover its use is subjected to the registration to a central authority, i.e. the registrant.

Building the citation graph requires the analysis of all the citation strings within each paper in order to locate the cited document. We have different scenarios depending on how the citing function T looks for the target work. If we choose to use URNs that can be simply retrieved by parsing the reference string, we can build the citation graph by analyzing each document at most once. If some of the cited document has not yet been known to the system, we simply put a placeholder which could be turned into an active work if the system will encounter it. Conversely, if the citing funtion needs to look for the cited work by perform a search in a database, it is quite likely that it is unable to match the description with one of the known works. For that reason we need to look again for the missed references each time a new document is submitted to the system.

Implementation For the sake of ease we identify each scholarly work by using a non-standard URN (i.e. in the sense that it does not use registered name-spaces) that is built by using the SHA-1 checksum of the PDF file. The main drawback of this solution is that since we are assigning names to the medium (i.e. the PDF file) rather to the scholarly work, we are prone to treat different representations of the same work as different ones. As a result the citation graph could contain more nodes than the correct one.

We rely on the reference strings in order to identify the cited works by a given paper. By using these design choices, the citing function T needs to perform a search within the database of the known works in order to identify if one of them corresponds to the given description.

As encoded by the function lookForCitedPaper in the algorithm ?, we adopt a filter and refine strategy in order to identify the work described by a given citation string. First we select a small set, which contains at most K elements, of candidate papers that have their title similar to the title field provided by the reference string. This is done by performing a keyword search on the database of the known documents using as query string all of the title's words field within the reference string. We aims to retrieve the K nearest neighbours in terms of the similarity to the given description.

We rely on zone indexing in order to distinguish the title from other zones of the document (cfr. section 3.1.6). Document relevance are computed relying on the equation 3.7 by giving an high value to the α coefficient which weights the title's index (i.e. $\alpha \in [0.85, 0.95]$). We choose to give a little degree of importance also to others document's zones in order to deal with parsing errors within the title field of the reference string.

Once we selected this set of candidate documents (which should be the K-nearest neighbours, in terms of similarity, to the given description), we evaluate

Algorithm 1 Citation graph building by using reference strings

```
1: procedure BuildCitationsGraph
2:     for all document do
3:         for all referenceString ∈ document do
4:             citedDocument ← LookForCitedPaper(citationString)
5:             if citedDocument then
6:                 append citedDocument to document.references
7:                 append document to citedDocument.citedBy
8:             end if
9:         end for
10:    end for
11: end procedure

12: function lookForCitedPaper(referenceString)
13:    filterTopK ← computeTopK(referenceString)
14:    refineSet ← {}
15:    for all document ∈ filterTopK do
16:        CheckAttributes(document)
17:        update refineSet
18:    end for
19:    describedDocument ← order1(refineSet)
20:    return describedDocument
21: end function
```

the similarity degree between those documents and the description provided by the reference string by using a scoring function. We take into account authors, the number of common words between the document's title and the title field of the reference string. Because the fields within the reference string could be affected by errors, that could be due both by the author or by the information extraction subsystem, we rely on some techniques such as the computation of the edit distance between the strings we are going to compare. The edit distance between two strings of characters is the number of operations required to transform one of them into the other. We use the metric defined by the Levenshtein distance that is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. If one of the document within the set of candidate is evaluated by the scoring function above a certain threshold, we assume the document as the target of the description given by the reference string and we mark the corresponding reference as *"shot"*.

Building the citation graph of the entire document's collection requires the analysis of all the references within each document. Because the citing function T could not be able to match some citations with a target document, we should

	Known Items	
	Relevant	Non Relevant
Retrieved	true positives (t_p)	false positives (f_p)
Not Retrieved	false negatives (f_n)	true negatives (t_n)

Table 2.2: The table of confusion (also known as confusion matrix) summarizes predictions on a set of known examples.

analyze again all of them whenever a new document joins the collection.

2.5 Prototype evaluation

As we described in section 2.3.1, the task of labeling the text chunks within a document D according to certain pre-defined data types, partitions the set D into two subsets, i.e. the subset of *relevant* and the subset of *not relevant* tokens with respect to a given data type. The partition of the set D is induced by the equivalence relation $\sim$ which specifies how to assign a data type k_j to each token t_i.

Table of confusion The automated information extraction system relies on an approximation $\sim_a$ of the equivalence relation $\sim$, which induces a different partition on the set D into the subset of the *retrieved* and the subset of *not retrieved* tokens. In order to assess the extent of the approximation of $\sim_a$, and therefore the effectiveness of the automated information extraction task, we can evaluate the degree of the overlap between those different partitions induced on D. By taking the Cartesian product of those two partition we can identify a new partition of D made up of four subsets, as depicted in table 2.2. Each element t_i of the set D belongs to one of the subsets depending if its known type is predicted correctly.

By analyzing how elements t_i are mapped into each block we can evaluate the effectiveness of the classifier, i.e. how much the approximation $\sim_a$ is close to the ideal $\sim$.

2.5.1 Effectiveness metrics

Different measures [MRS08] are used in order to summarize the content of the confusion matrix. This because different problem domains call for the need to use different measures for summarizing prediction quality. We are going to review these measure and their scope of usage, then we evaluate our prototype.

Accuracy *Accuracy* is the fraction of predictions that are correct. Since a classifier attempts to label instances as neither relevant or non relevant according to a given data type, the accuracy gives a measure of how many elements t_i are properly

labeled considering both the true negatives t_n and the true positives t_p.

$$\text{accuracy} = \frac{|t_p + t_n|}{|t_p + f_p + f_n + t_n|} \tag{2.2}$$

Accuracy is not an appropriate measure in all of circumstances in which the data are extremely skewed[11]. In these situations most of the elements t_i are mapped into the true negatives t_n subset. A system tuned to maximize accuracy can appear to perform well as long as the classifier is not predicting too many positives or just by simply label all the item t_i as non relevant (i.ewith no items in the true positive t_p subset). However labeling all the items t_i as non relevant is completely unsatisfying. The accuracy will always be high

The measures of precision and recall concentrate the evaluation on the return of true positive (t_p), asking what fraction of the relevant documents have been found and how many false positives have also been returned.

Precision *Precision* is the fraction of positive predictions that are correct.

$$\text{precision} = P(\text{retrieved}|\text{relevant}) = \frac{|t_p|}{|t_p + f_p|} \tag{2.3}$$

Recall *Recall* (also called sensitivity) is the fraction of positive labeled instances that were predicted as positive. Note that an high level of recall can be easily achieved by simply labeling all the items t_i as true positives t_p. This because recall does not take into account the number of false positives f_p since they affects the precision measure.

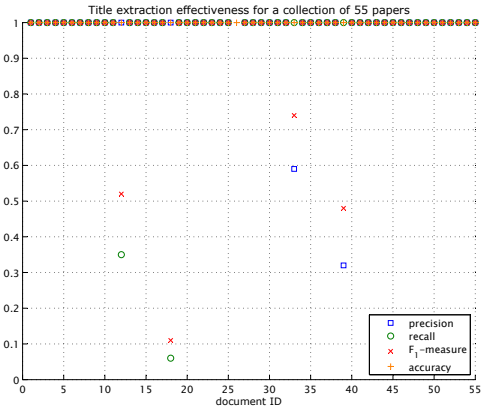
$$\text{recall} = P(\text{retrieved} \mid \text{relevant}) = \frac{|t_p|}{|t_p + f_n|} \tag{2.4}$$

F-measure A single measure that trades off precision versus recall is the F-measure, which is the weighted harmonic mean of precision and recall (equation 2.5a). Values of $\beta < 1$ emphasize precision, whereas values of $\beta > 1$ emphasize recall.

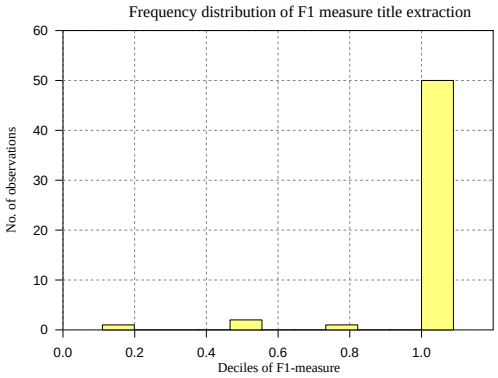
$$F = \frac{1}{\frac{\alpha}{\text{precision}} + \frac{1-\alpha}{\text{recall}}},\ \alpha \in [0, 1] \tag{2.5a}$$

$$\beta^2 = \frac{1-\alpha}{\alpha} \Rightarrow F = \frac{(\beta^2 + 1)\,\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{2.5b}$$

[11] *Skewness* is a measure of the asymmetry of the probability distribution of a random variable. The probability distribution of skewed data are not symmetrical about the mean.



(a) Each point in the horizontal axis stands for a paper. For each of them we plotted the effectiveness of the title extraction by using four different metrics. All of them was computed by using a granularity of single chars.



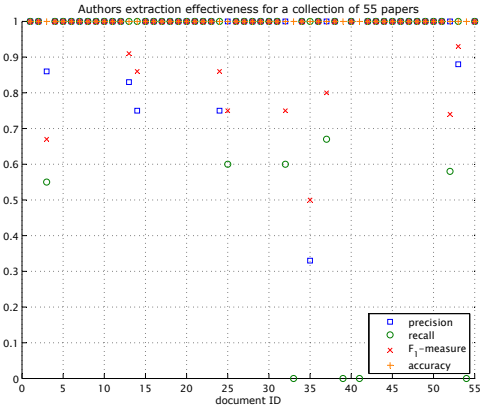
(b) Statistical distribution of the F_1-measure deciles within the test collection.

Figure 2.6: Effectiveness of the title extraction for a test collection of 55 papers. Most of the title have been extracted correctly. Errors are due to differences while using text typesets in title formatting.

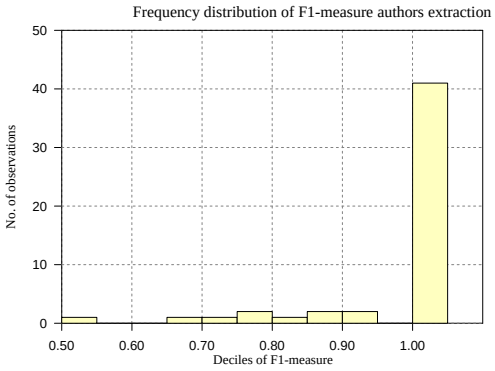
The default balanced F-measure equally weights precision and recall, which means making $\beta = 1$. In this case the equation 2.5b simplifies to equation 2.6.

$$F_{\beta=1} = \frac{2\,\text{precision}\,\text{recall}}{\text{precision} + \text{recall}} \tag{2.6}$$

The harmonic mean is used rather than the simply average (i.e. the arithmetic mean), which is less suitable for weighting both precision and recall. For instance suppose than we achieve an high level of recall by simply labeling all the instance t_i as relevant. As instance if we assume that 1 item in $10,000$ is relevant to a given data type, we yield precision $= 10^{-3}$. By using the arithmetic mean we yield a value of $F_1 \approx 50\%$. In contrast if we use the armonic mean by equally weight precision and recall, the strategy is scored as $F_1 \approx 0.02\%$. The harmonic mean is always less than or equal to the arithmetic mean and the geometric mean, hence when the values of two numbers differ greatly, the harmonic mean is closer to their minimum than to their arithmetic mean.

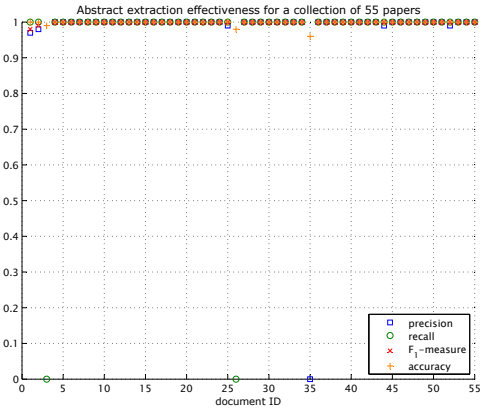


(a) Each point in the horizontal axis stands for a paper. For each of them we plotted the effectiveness of the authors' string extraction by using four different metrics. All of them was computed by using a granularity of single chars.

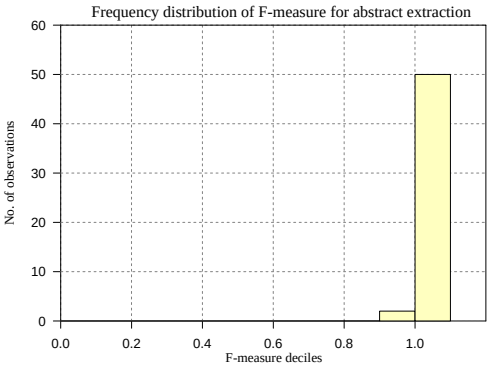


(b) Statistical distribution of the F_1-measure deciles within the test collection.

FIGURE 2.7: Effectiveness of the authors' extraction for a test collection of 55 papers. Errors are due to difficult in distinguishing author's name to other information (e.g. author's affiliation) within the front matter.

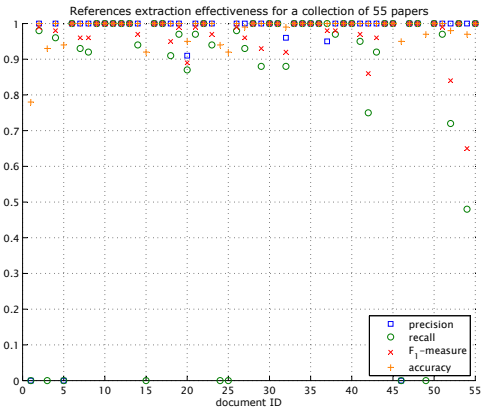


(a) Each point in the horizontal axis stands for a paper. For each of them we plotted the effectiveness of the abstract section extraction by using four different metrics. All of them was computed by using a granularity of single chars.

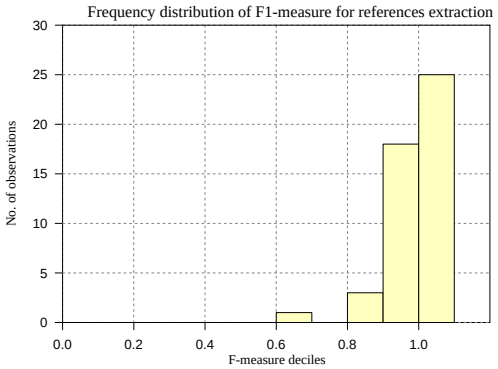


(b) Statistical distribution of the F_1-measure deciles within the test collection.

FIGURE 2.8: Effectiveness of the abstract section extraction for a test collection of 55 papers. Most of the papers in our dataset distinguish the abstract my make usage of keywords. However in those cases for which different rules are used, our algorithm fails.

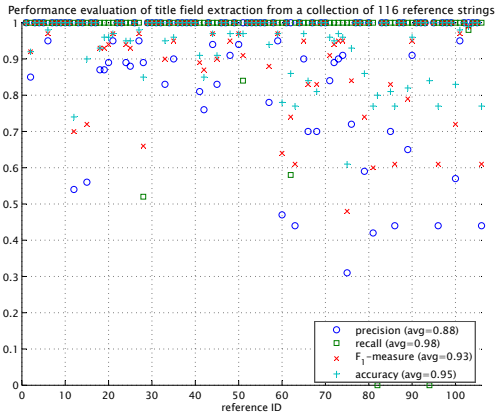


(a) Each point in the horizontal axis stands for a paper. For each of them we plotted the effectiveness of the reference strings extraction by using four different metrics. All of them was computed by using a granularity of single chars.

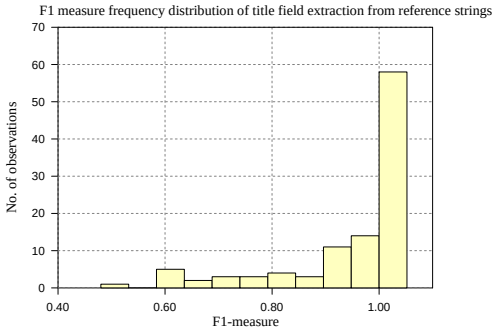


(b) Statistical distribution of the F_1-measure deciles within the test collection.

Figure 2.9: Effectiveness of the reference strings extraction for a test collection of 55 papers. Most of the errors comes for different styles which are used for delineating individual reference strings. Our prototype is capable to split citations separated with delimiters. The extent of reference extraction effectiveness affects the citation graph building which relies mainly on reference strings.



(a) Each point in the horizontal axis stands for a reference string. For each of them we plotted the effectiveness of the title field labeling by using four different metrics. All of them was computed by using a granularity of single chars. We obtain an average F_1-measure of 0.93 as stated in [CGK08] for the CiteSeer^X dataset.



(b) Statistical distribution of the F_1-measure deciles within the test collection.

Figure 2.10: Effectiveness of the title field labeling within reference strings by using the ParsCit-080402 reference string parsing package [CGK08]. The test collection is made up of 106 reference strings.

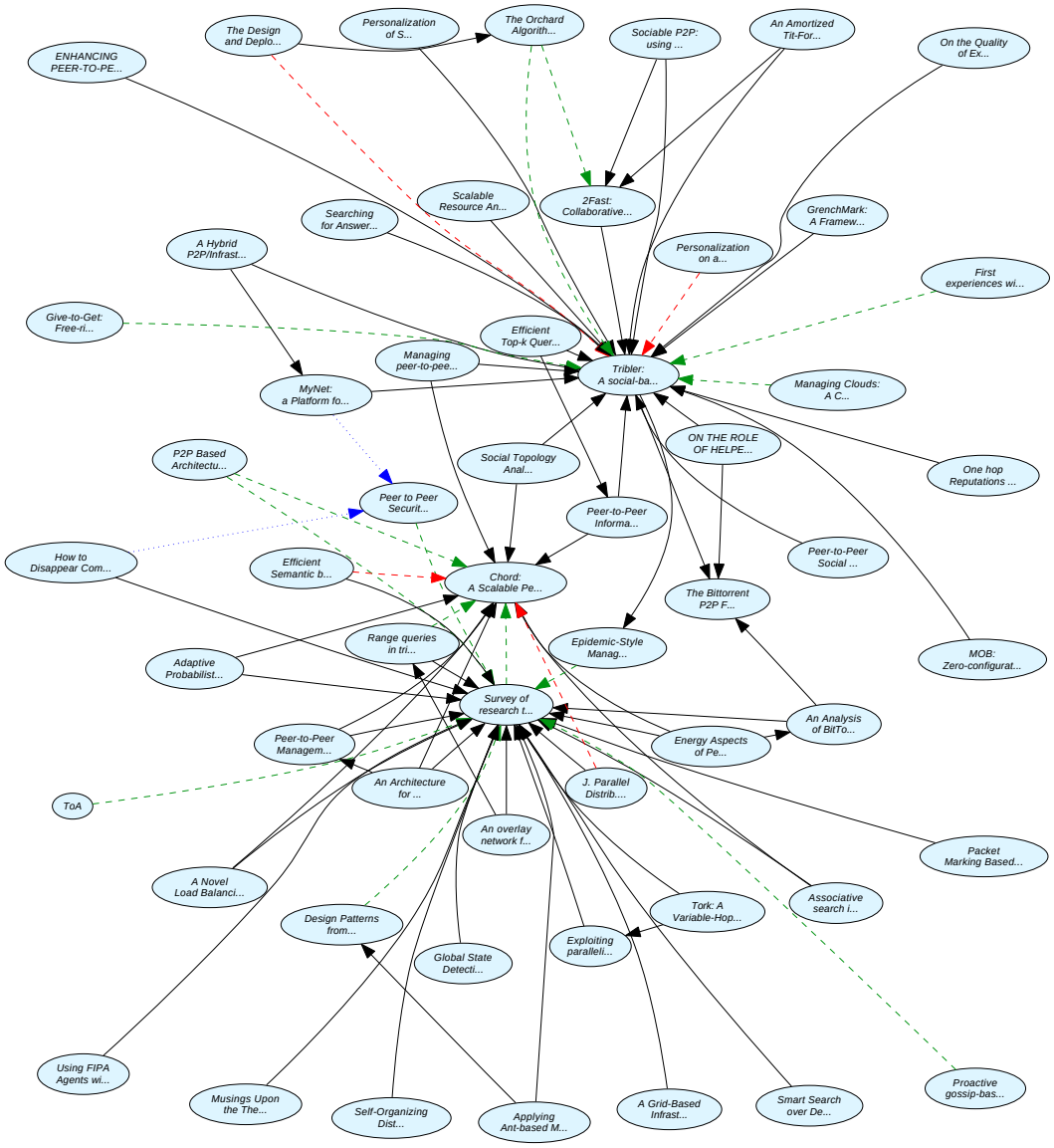


FIGURE 2.11: Effectiveness of proposed algorithm while building citation graph over our test collection. It relies on a filter&refine strategy which make usage of reference strings extracted from each paper. Dotted blue lines depict false positives: they are due to 1) inaccuracy while extracting and segmenting the reference strings; 2) inaccuracy of the bibliographic data of the target reference 3) lack in checking the reference string field with the target properties. Our prototype takes into account only the words within the title. Green and red arrows depict both false negatives, but the while the former are due to lack of reference string data extraction (recall=0), the latter are due to refine stage inaccuracy.

	ds	t_p	f_p	f_n	t_n	Precision	Recall	F_1	Accuracy
$k = 3$	D	39	2	43	3,165	0.95	0.48	0.63	0.99
	D_r	39	2	28	3,180	0.95	0.58	0.72	0.99
$k = 5$	D	64	2	19	3,164	0.97	0.77	0.86	0.99
	D_r	64	2	4	3,179	0.97	0.94	0.96	1.00

Table 2.3: Summary of the citation graph building algorithm effectiveness. Our algorithm adopts a *filter&refine* strategy, in which the k-size filter is filled by the top-k results of a keyword search algorithm which uses as keywords the words from paper's title. Size k of the filter stage affects the effectiveness of the linking algorithm. A wide filter ensures better recall although it requires more computational resources. We assessed the quality of our algorithm for $k = 3$ and $k = 5$ by using our test collection D. Since the reference string extraction failed (i.e. recall $= 0$) for some items, we repeated the experiments by using the test collection D_r made up by excluding those items from D.

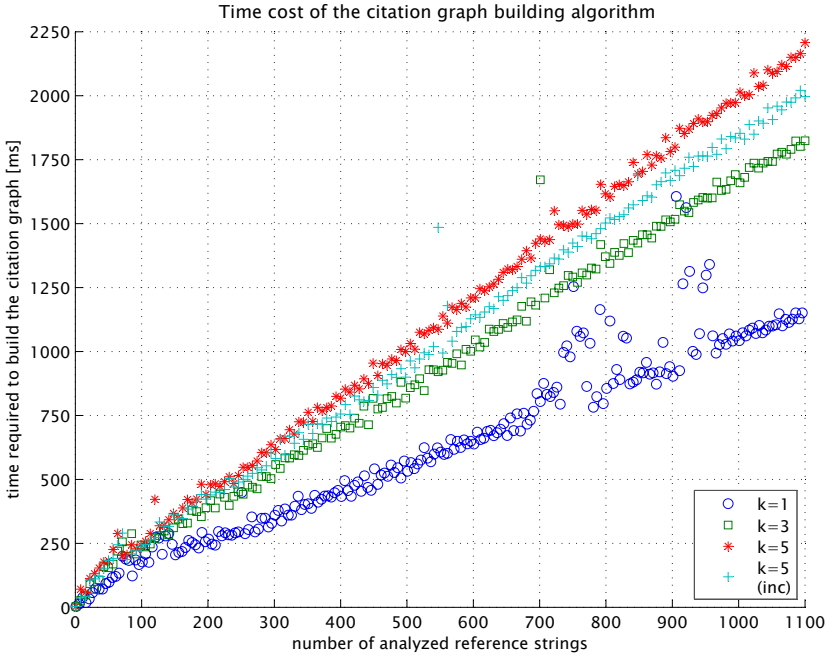


Figure 2.12: Evaluation of the time cost required to build the citation graph. As pointed out in table 2.3 larger filter size k ensures higher recall, although it requires more computation steps since each item in the filter needs to be analyzed by the refine algorithm. Data labeled as $k = \{1, 3, 5\}$ assesses cost for naive algorithm which checks each citation string in the database. Speed-up can be obtained by avoiding those citations which have been already linked (cf. $k = 5_{inc}$). All the experiments was done by using `CPython 2.6.2` byte-code interpreter on Linux kernel `2.6.28-12-generic` running on Intel Mobile Pentium 4-M 1.4GHz processor.

Chapter 3

Models and techniques for linked text collections analysis

In chapter 2 we defined a data model for scientific literature in which a paper is considered basically as a container of text zones, each of them with an associated semantics. Moreover papers are strongly connected each other by citations.

This chapter focuses on models and techniques which have been traditionally employed while analyzing text documents with links between them, such as Web pages. A retrieval model should resort on these techniques while modelling the notion of document relevance with respect to user's information needs.

Section 3.1 discusses about the Vector Space Model which relies on a statistical model in order to map text into vectors leading to a huge term-by-document matrix. By performing linear algebraic operations on this matrix, semantics relationships between documents can be better highlighted. Section 3.2 treats about this technique which is known as Latent Semantic Analysis. Section 3.3 discusses about usefulness of clustering together related documents. Section 3.4 discusses about link analysis which aims to exploit information which lies onto the topological structure of the directed graph induced by citations between papers.

3.1 The Vector Space Model

In the Vector Space Model (VSM) [BDJ99] each document in a collection is represented as a vector in a $\mathbb{R}^t$ vector space. Each component of the vector reflects a particular concept, keyword or term associated with the given document. The value assigned to that component reflects the importance of the term in representing the semantics of the document. Typically, that value is a function of the frequency with which the term occurs in the document or in the whole collection.

A database containing a total of n documents described by m terms is represented by a $m \times n$ term-by-document matrix $\mathbf{A}$ (3.1). The n vectors representing the n documents form the columns of the matrix. Thus, the matrix element w_{ij}

gives a measure on how well the term i represents the semantics of the document doc_j.

$$\mathbf{A} = \begin{array}{c} \\ \text{term}_1 \\ \text{term}_2 \\ \vdots \\ \text{term}_m \end{array} \begin{array}{c} \begin{array}{cccc} \text{doc}_1 & \text{doc}_2 & \cdots & \text{doc}_n \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix} \end{array} \tag{3.1}$$

We refer the columns of **A** as *document vectors* and the rows of **A** as *term vectors*. The semantic content of the database is wholly contained in the column space of **A**, meaning that the document vectors span the content. Geometric relationships between document vectors can be exploited to model similarities and differences in content.

3.1.1 Stop-listing and word stemming

The choice of terms used to describe the database determines not only its size but also its utility. Including very common terms like *to* or *the* would do little to improve the quality of the term-by-document matrix. The process of excluding such high-frequency words is known as *stoplisting*.

In constructing a term-by-document matrix, terms are usually identified by their word stems. For example, the word *communities* counts as the term *community* and the word *downloading* counts as the term *download*. Stemming reduces storage requirements by decreasing the number of words maintained. We use the Porter' stemming algorithm for this purpose [Por97].

3.1.2 The tf-idf weighting scheme

A variety of schemes are available for weighting the matrix elements w_{ij}. The tf-idf weighting scheme assigns to each element a weight that depends on two factors, as defined in (3.2).

$$w_{ij} = \text{tf}_{ij} \cdot \text{idf}_i \tag{3.2}$$

Term frequency The term frequency $\text{tf}_{i,j}$ is a local weight that reflects the importance of a term t_i within the document doc_j. A document doc_j which mentions a term t_i more often than a document doc_k should receive an higher score with respect to the term t_i.

Inverse document frequency The inverse document frequency idf_i is a global weight that reflects the overall value of term t_i as an indexing term for the entire collection. Terms common to many documents give less information about the

semantic content of a document. As one example, consider a very common term like *computer* within a collection of papers on *computer science*. Including that term in the description of a document gives little information about its content. The idf_i factor takes into account the discriminative power of the term t_i within the collection and it is negatively correlated with the number of documents N_i in which t_i appears. It can be computed as defined in (3.3), where N is the total number of document in the collection. According to this definition, a term present in each document of the collection as no discriminative power, that is $\text{idf}_i = 0$.

$$\text{idf}_i = \log \frac{N}{N_i}, \quad N_i = |\{\text{doc}_j : t_i \in \text{doc}_j\}| \tag{3.3}$$

The tf-idf weighting scheme assigns to term t_i a weight w_{ij} with respect to the document doc_j that is: highest when t_i occurs many times within a small number of documents; lower when t_i occurs fewer times in a document or occurs in many documents; lowest when the term occurs in virtually all documents.

3.1.3 Documents similarity

For a document doc_j, the set of weights may be viewed as a quantitative digest of the document. According to this view, known as *bags of words model*, the exact ordering of the terms is ignored, but we only retain statistical information about the frequency of occurrence of each term. This model assumes that two documents with similar bags of words representations are close in their semantic content.

The VSM proposes to evaluate the similarity of two documents by measuring the correlation of the corresponding document vectors. A common way to measure correlation is computing the cosine of the angles between the two vectors, as defined in (3.4).

$$sim(\mathbf{doc_j}, \mathbf{doc_k}) = \cos\theta_{j,k} = \frac{\mathbf{doc_j} \cdot \mathbf{doc_k}}{\|\mathbf{doc_j}\|\|\mathbf{doc_k}\|} = \frac{\sum_i w_{i,j} \cdot w_{i,k}}{\sqrt{\sum_i w_{i,j}^2}\sqrt{\sum_i w_{i,k}^2}} \tag{3.4}$$

Because the query and document vectors are typically sparse, the dot product and norms in (3.4) are generally inexpensive to compute. Furthermore, the document vector norms need to be computed only once for any given term-by-document matrix. Note that multiplying either $\mathbf{doc_j}$ or $\mathbf{doc_k}$ by a constant does not change the cosine value. Thus document vectors may be scaled by any convenient value. Because the content of a document is determined by the relative frequencies of the terms and not by the total number of times particular terms appear, the matrix elements in (3.1) can be scaled so that the Euclidean norm of each column is 1, that is $\|\text{doc}_j\| = 1$. With this choice the similarity computed with (3.4) gives the geometric distance between a document vector pair.

3.1.4 Query representation

When a user is wishing to retrieve some information from the collection, he queries the system by describing his information needs. This can be done by providing a set of terms, perhaps with weights, which aims to reflect the semantics of his information need. According to the VSM, the query can be represented just like a document. It is likely that many of the terms in the database do not appear in the query, meaning that many of the query vector components are zero.

$$\mathbf{q} = \begin{pmatrix} q_1 & \cdots & q_t \end{pmatrix}^T \tag{3.5}$$

Query matching is finding the documents most similar to the query. In the VSM the documents selected are those geometrically closest to the query, according to the similarity measure defined in (3.4). The search for relevant document is carried out by computing the cosines of the angles θ_j between the query vector $\mathbf{q}$ and the document vectors doc_j. Relevant documents can be ranked according to their similarity with the query.

3.1.5 Inverted index and posting files

As a document generally uses only a small subset of the entire dictionary of terms generated for a given database, most of the elements w_{ij} of a term-by-document matrix are zero. Storing the matrix in a naive way results in a wasteful use of the memory. A much better representation is to record only the non-zero values. The commonest solution adopted in text retrieval systems is a data structure known as *inverted index*. Basically it is a list containing all the t different terms of the collection, so it has as many items as the number of rows in (3.1). For each term we maintain a list, known as *posting list*, which stores the weights only for the documents in which the given word occurs. By using this solution we store the same information retained by the term-by-document matrix avoiding the zero entries.

When a user queries the system, the search for relevant document is carried out by retrieving the posting lists of the query terms and taking their intersection in order to compute the cosine similarities as defined in (3.4).

Example Consider a collection made up of n documents and a vocabulary of m terms. Assuming that each entry w_{ij} in the term-by-doc matrix requires k bytes, $n \cdot m \cdot k$ bytes are required to store the whole matrix in the memory. As instance,

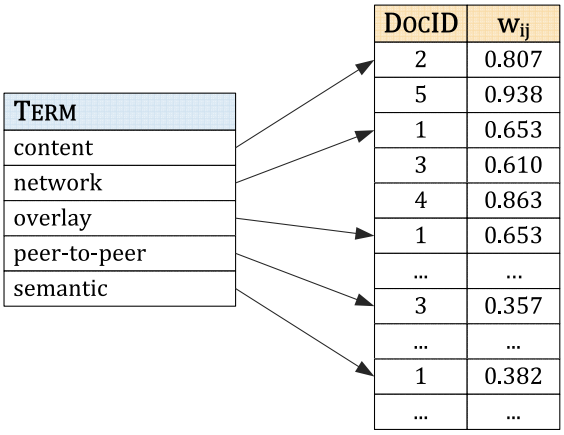


FIGURE 3.1: Example of using the inverted index and posting files data structure for representing large sparse matrix. The content of the term-by-document matrix in (3.6) can be represented by storing each term as an entry in the inverted index, each of them pointing to a *posting list* of non-zero values. Each entry in the posting lists refers to a document doc_j. This strategy avoids wasting of resources due on storing zero entries, although it requires to store the document *id* and m pointers for each entry. Therefore this representation becomes useful as increasing sparsity of the term-by-document matrix.

the collection represented by the equation (3.6) requires $30 \cdot k$ bytes.

$$\mathbf{A} = \begin{array}{l} \\ \text{content} \\ \text{network} \\ \text{overlay} \\ \text{peer-to-peer} \\ \text{semantic} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{doc}_1 & \text{doc}_2 & \text{doc}_3 & \text{doc}_4 & \text{doc}_5 & \text{doc}_6 \end{array} \\ \begin{pmatrix} 0 & 0.807 & 0 & 0 & 0.938 & 0 \\ 0.653 & 0 & 0.610 & 0.863 & 0 & 0 \\ 0.653 & 0.509 & 0.610 & 0 & 0 & 0 \\ 0 & 0 & 0.357 & 0.505 & 0.346 & 0.707 \\ 0.382 & 0.298 & 0.357 & 0 & 0 & 0.707 \end{pmatrix} \end{array} \tag{3.6}$$

The same content could be stored by relying on the inverted index and posting files data structures, as depicted in figure 3.1, which avoids storing zero entries. However it requires to store document identifiers, which we suppose to require h bytes, and n pointers each of them tooks p bytes. This representation become advantageous as increasing sparsity of the matrix $\mathbf{A}$.

3.1.6 Zone indexing

The VSM treats a document as an unstructured sequence of terms. It aims to model the semantics of a document by using statistical informations about words occurrences. However most documents have additional structure. Text within human readable document often comes with typographic information. As example, a word written in bold font should be considered much more important since

the author emphasized it. Moreover the text content could be organized within a number of paragraphs that may have different importance. For instance, with respect to our case study, the abstract section within a scholarly work, which summarizes paper's content, can convey better the semantics of the document. The words chosen for the title can be very important as well. In order to improve the notion of relevance we can exploit also the structure within the document.

Because we are dealing with scholarly literature, we treat each document as a container of predefined text zones, such as the title, the abstract and the body. We can represent each zone with its zone vector and store it as column in the matrix (3.1). Tf-idf weight computation will refer to zone instead of document in order to compute the global *idf* factor (3.3).

Instead of use a unique term-by-document matrix we use multiple term-by-zone matrix, each of them charged for a specific zone. A much more efficient solution could be using a unique inverted index encoding the zones directly in the posting lists.

The relevance of a document to a given query can be computed by calculating the linear combination of the relevance scores of each zone (3.7). Coefficient are properly chosen in order to reflect the importance of each section within the document.

$$sim(\mathbf{q}, \mathbf{doc_j}) = \alpha\, sim(\mathbf{q}, \mathbf{title_j}) + \beta\, sim(\mathbf{q}, \mathbf{abs_j}) \tag{3.7}$$

For instance, we can consider finding the word *tribler* within the title much more important than finding it within the body. We can also support queries as searching for documents which contains a certain word in a certain section of the document, e.g. search for documents that contains the word *tribler* within the title.

3.1.7 Implementation issues

Algorithm 2 Global Inverse Document Frequency computation algorithm

```
1:  procedure updateIdf(dictionary, N)          ▷ N documents in the collection
2:      for all term ∈ dictionary do
3:          postingFile ← dictionary[term]
4:          N_i ← |postingFile|          ▷ Number of documents in which term occurs
5:          idf ← log N/N_i
6:          for all docId ∈ postingFile do
7:              (t_f, w_ij) ← postingFile[docID]
8:              w_ij ← t_f · idf
9:              postingFile[docID] ← (t_f, w_ij)
10:         end for
11:     end for
12: end procedure
```

Algorithm 3 Document vectors normalization algorithm

```
1: procedure NORMALIZEDOCUMENTVECTORS(dictionary)
2:     docWeights ← {}
3:     for all term ∈ dictionary do
4:         postingFile ← dictionary[term]
5:         for all docId ∈ postingFile do
6:             (t_f, w_{i,j}) ← postingFile[docId]
7:             docWeights[docId] ← docWeights[docId] + w_{ij}^2
8:         end for
9:     end for
10:    for all term ∈ dictionary do
11:        postingFile ← dictionary[term]
12:        for all docId ∈ postingFile do
13:            (t_f, w_{ij}) ← postingFile[docID]
14:            w_{ij} ← w_{ij} / √docWeights[docId]
15:        end for
16:    end for
17: end procedure
```

Algorithm 4 Computation of the k best documents

```
1: function COMPUTETOPKDOCUMENTS(query)
2:     resultSet ← {}
3:     for all term ∈ query do
4:         if term ∈ dictionary then
5:             postingFile ← dictionary[term]
6:             for all docId ∈ postingFile do
7:                 (t_f, w_{ij}) ← postingFile[docId]
8:                 resultSet[docId] ← resultSet[docId] + w_{ij}
9:             end for
10:        end if
11:    end for
12:    topK ← ORDERK(ResultSet)
13:    return topK
14: end function
```

3.2 Latent Semantic Indexing

In the Vector Space Model each term is modelled as a dimension in a $\mathbb{R}^t$ vector space where t is the number of different terms which appears in the corpus. This result in a high dimensional vector space. This scheme suffers from synonyms and noise in documents and this may results in a lack on capturing the semantics of documents. By making usage of linear algebraic operation over the term-by-doc matrix $\mathbf{A}$ (cf. equation 3.1) similarities between terms can be revealed since they are expected to occurs in semantically close documents. One way to reveal the rank of $\mathbf{A}$ is to compute its Singular Value Decomposition (SVD) [BDJ99, Cha03].

3.2.1 Singular Value Decomposition

SVD transforms a high dimensional vector, such as a document vector $\mathbf{A_j}$ of $\mathbf{A}$, into a lower-dimensional semantic vector by projecting the former into a semantic subspace. Each element of a semantic vector corresponds to the importance of an abstract concept. SVD decomposes $\mathbf{A}$ into the product of three matrices as described in equation 3.8 where r is the rank of $\mathbf{A}$, matrices $\mathbf{U}$, $\mathbf{V}$ are column orthonormal and the diagonal matrix Σ can be organized such that equation 3.8c holds.

$$\mathbf{A_{m\times n}} = \mathbf{U_{m\times r}} \underbrace{\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{pmatrix}}_{\Sigma_{r\times r}} \mathbf{V^T_{r\times n}} \tag{3.8a}$$

$$\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I} \tag{3.8b}$$

$$\sigma_1 \geq \cdots \geq \sigma_r \tag{3.8c}$$

The i^{th} row of $\mathbf{A}$ may be regarded as a n-dimensional representation of term t_i, just as the j^{th} column of $\mathbf{A}$ as the m-dimensional representation of document doc_j.

The i^{th} row of $\mathbf{U}$ is a refined representation of term t_i and the j^{th} column of $\mathbf{V}$ is a refined representation of document doc_j. Both are vectors in a r-dimensional subspace.

The standard cosine measure similarity between documents can be applied to the $\mathbf{A}$ matrix: entries of $(\mathbf{A}^T\mathbf{A})_{n\times n}$ may be interpreted as the pairwise document similarities in vector space. Symmetrically we can regard the entries of $(\mathbf{A}\mathbf{A}^T)_{m\times m}$ as the pairwise term similarity based on their co-occurrences in documents.

Since $\mathbf{A}$ has redundancy revealed by SVD, we can compute documents pairwise similarities as $\mathbf{V}\Sigma^2\mathbf{V}^T$ and terms pairwise similarities as $\mathbf{U}\Sigma^2\mathbf{U}^T$.

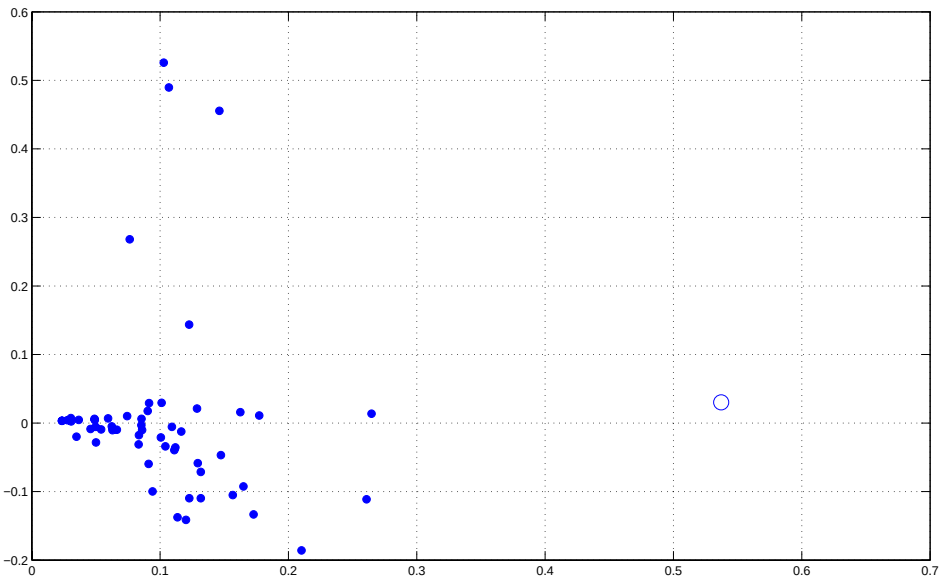


FIGURE 3.2: Projection on a 2-dimensional subspace of documents vectors (rows of $\mathbf{V}_2^T$) from our test collection. Since LSI maps similar documents into similar vectors, closest points have similar contents. Note that the projection of the document vectors' centroid does not represents the average semantics of the whole collection.

3.2.2 Low rank approximation

Latent Semantic Indexing (LSI) approximates the matrix $\mathbf{A}$ of rank r with a matrix $\mathbf{A_k}$ of lower rank k by retaining only the largest k singular values out of r, i.e. $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq \ldots \geq \sigma_r$.

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T \tag{3.9}$$

By choosing an appropriate k the important structure of the corpus is retained while the noise and variability in word usage, which is related to smaller σ_i, is eliminated. Studies on LSI suggest setting $k \in [50, 350]$ [DDF+90].

LSI is capable of bringing together documents that are semantically related even if they not share terms, by learning from co-occurring word usage. Symmetrically it can discover polysemy and synonymity of terms.

3.2.3 Examples

LSI maps similar terms into close vectors exploiting correlation between terms. Symmetrically it maps similar documents into close vectors. By retaining just $k = 2, 3$ singular values, term vectors and document vectors can be plotted. As example we considered our test collection, which is the same considered in fig-

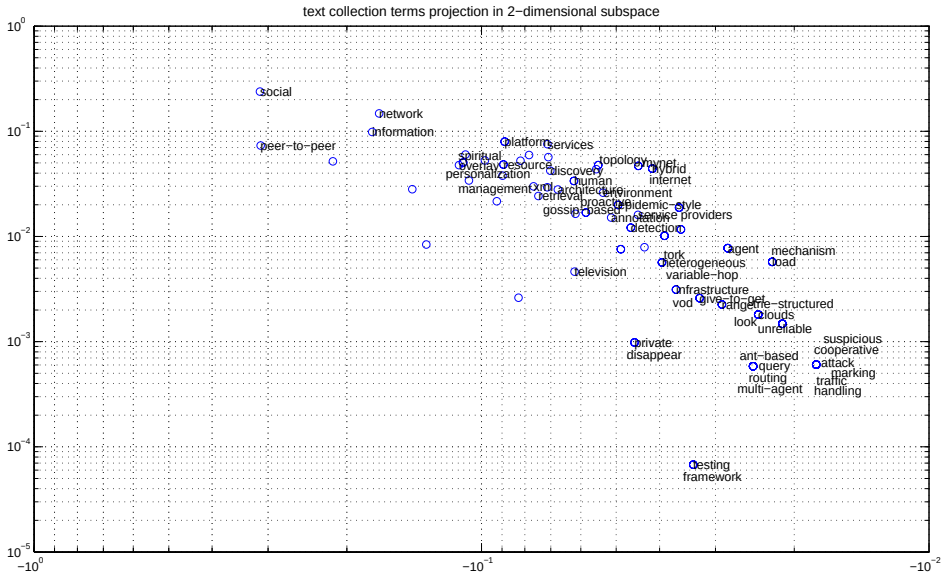


Figure 3.3: Projection on a 2-dimensional subspace of term vectors (rows of $\mathbf{U}_2$) from our test collection. We take into account just words which forms titles. Some words have been removed for clarity.

ure 2.11, and we performed the LSI by retaining just $k = 2$ singular values on the term-by-doc matrix $\mathbf{A}$, which has built by using the Vector Space Model for just the words which forms the title for each paper.

We resort to the Matlab command `[U,S,V]=svds(A,K)`, where `K` is the number of singular values of `A` to be retained, which computes the approximated matrix as described in equation (3.9).

In figure 3.2 is depicted the projection on a 2−dimensional subspace of the document vectors, i.e. the rows of the approximated matrix $\mathbf{V}_2^T$. Since LSI maps exploit correlations between documents, it maps similar documents to similar vectors. Two-dimension Cartesian space may be interpreted as a semantics space in which a notion of distance in semantics between points can be considered. A collection of documents, resulting in a certain distribution over the semantic space may be summarized by a unique point, which may be the average, which gives a measure of the whole collection' semantics. Note that this is not the same of the average document vectors computed from the VSM matrix $\mathbf{A}$ which does not take into account correlation between terms, but treat all of them referring to orthogonal dimensions.

In figure 3.3 is depicted the projection on a 2−dimensional subspace of the term vectors, i.e. the rows of the approximated matrix $\mathbf{U}_2$.

3.3 Clustering

3.3.1 Problem statement

We are given a collection D of items which are characterized by an internal property which can be employed for defining a measure of distance between any two pairs of items. The clustering function γ partitions the collection D of items into k parts $\{D_1, \ldots, D_k\}$, i.e. $\gamma : D \mapsto \{D_1, \ldots, D_k\}$, so as to minimize the intra-cluster distance. Items within a cluster should be as close as possible while items in one cluster should be as far as possible from those items of other clusters.

Internal representation of items specify also the representation of the cluster. Assuming that each item may be represented as a vector $\mathbf{x}$ in a vector space, a cluster of items may be represented by the centroid $\vec{\mu}$ (average) of the items vectors:

$$\vec{\mu} = \frac{1}{|D_i|} \sum_{\mathbf{x} \in D_i} \mathbf{x} \tag{3.10}$$

A measure of how well the centroids represent the member of their clusters is the residual sum of squares i.e. the squared distance of each vector from its centroid summed over all vectors:

$$\mathrm{RSS}_i = \sum_{\mathbf{x} \in D_i} |\mathbf{x} - \vec{\mu}(D_i)|^2 \tag{3.11a}$$

$$\mathrm{RSS} = \sum_{1}^{k} \mathrm{RSS}_i \tag{3.11b}$$

Example We can consider as items the documents of a collection and we can use as internal property the Vector Space Model, which gives a measure of distance between any two pairs of document vectors. Similarly we may consider as internal property the low k-rank approximation of document vectors given by the Latent Semantic Analysis. In the former case we cluster documents in a high-dimension subspace, in the latter option we can refer to the semantic Cartesian space, such as the one in the figure 3.2 and cluster documents in this semantic space.

3.3.2 Clustering algorithms

Algorithm which employs the clustering function γ can follow two different paradigms [Cha03, MRS08].

Bottom-up approach of clustering, as known as agglomerative, considers each items in a group of its own and proceed by collapsing together groups of items until the number of partitions is suitable.

Top-down approach of clustering declares the number k of partitions and assigns items to partitions $\gamma : D \mapsto \{D_1, \ldots D_k\}$ in order to minimize the intra-cluster

distance of elements and maximize the distance between items which belongs to different clusters. $k-$means is an example of clustering algorithm which follows this paradigm.

3.3.3 Cluster hypothesis

The utility of clustering for information retrieval lies in the so-called cluster hypothesis: given a suitable clustering of a collection, if the user is interested in document doc_j he is likely to be interested in other members of the cluster D_i to which doc_j belongs, i.e. documents in the same cluster behave similarly with respect to relevance to information needs.

The cluster hypothesis is not limited to document alone. If documents are similar because they share terms, terms can also be represented as term-vectors representing the documents in which they occurs, and these term-vectors can be used to cluster the terms (cf. figure 3.3).

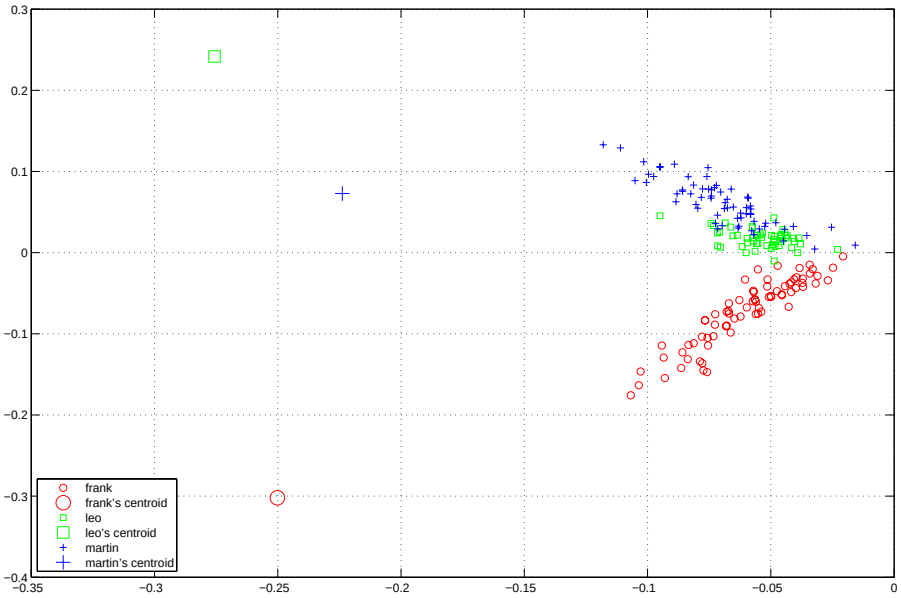
As with terms and documents, we can set up a bipartite relation for people liking documents and use this to cluster both people and documents, with the premise that similar people like similar documents and vice versa. This important ramification of clustering is called collaborative filtering.

Example In order to show the proof-of-concepts of the clustering hypothesis I collected the research paper which have been read by my colleagues while performing their researches. Each paper has been modelled as a vector by using the Vector Space Model. By performing a low rank reduction with the Latent Semantic Indexing, document vectors have been projected into a 2-dimensional subspace, which is intended as a semantic subspace according to the LSI capabilities.

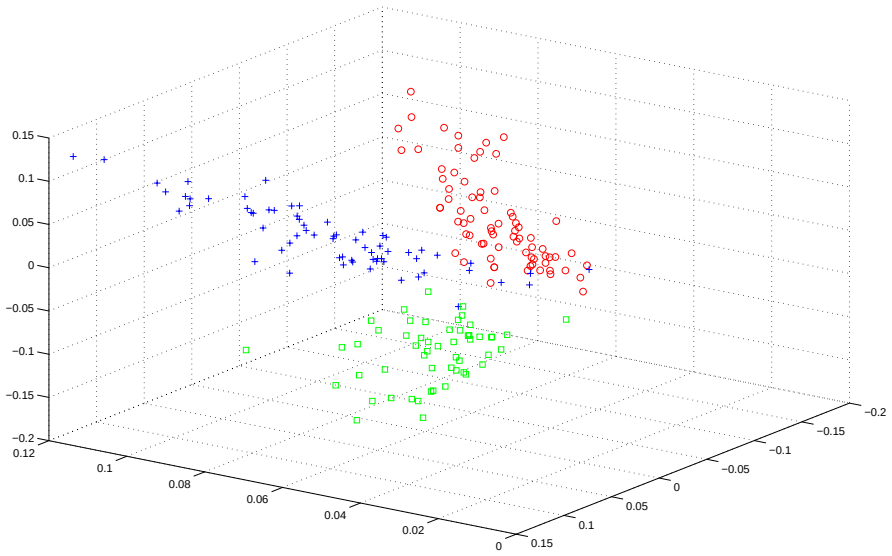
In figure 3.4 is depicted the projection of document vectors which have been obtained resorting on the Vector Space Model of the words which forms the abstract section of each document. Clearly documents which have been relevant for each user show clustering features according to users' research directions and preferences.

We can provide an insight of each user research preferences by computing the centroid of the document vectors. Centroid computed by considering the low-rank approximation of document vectors maps the average research direction as the average vector within the cluster into the semantic subspace. If we compute the centroids by using the high-dimension document vector we do not take into account correlations between words. As depicted in figure 3.4(a) these centroids does not corresponds to the semantic cluster centroid.

Maybe each user could find relevant those items which are relevant for other users which are closer to its research direction semantics. We can recommend to each user those items by computing the distance, i.e. the similarity, of those items which are closer to their centroid.



(a) Projection of document vectors into a 2-d semantic subspace, i.e. obtained by retaining the 2 largest singular values while performing LSI



(b) Projection of document vectors into a 3-d semantic subspace, i.e. obtained by retaining the 3 largest singular values while performing LSI. A further dimension allows a better distinction of users' clusters.

FIGURE 3.4: Proof-of-concepts of cluster hypothesis. We collected papers relevant for 3 different users while performing their researches. We used a Vector Space Model resorting just on words which comes from abstract section of each paper.

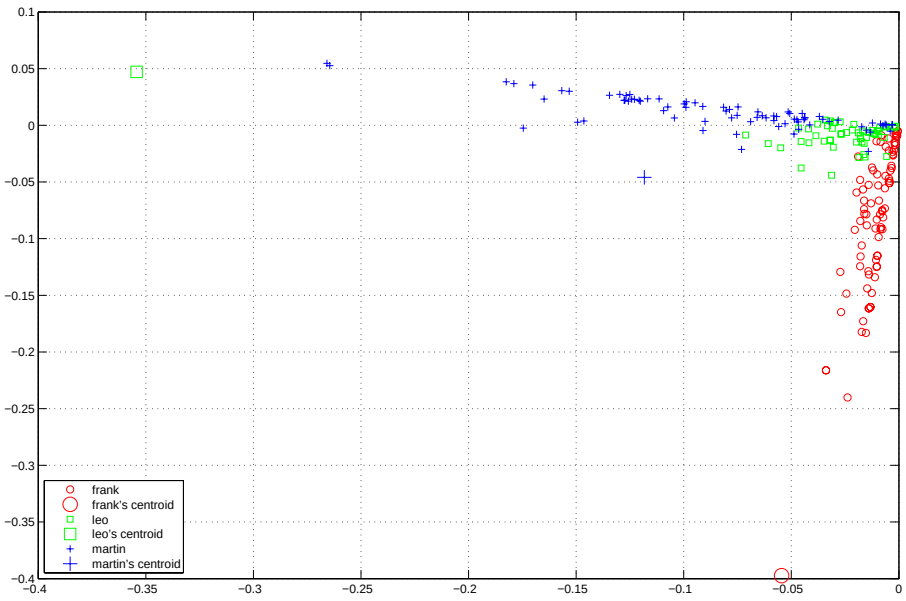


Figure 3.5: Projection on a 2-d semantic subspace of document relevant for 3 different users. Documents data set are the same of figure 3.4, but here the Vector Space Model used for document vectors refers just to words of the papers' titles.

3.4 Link analysis

Citations between semantically related work are an important feature of scientific literature since they encodes knowledge of the author about a portion of the scientific literature. By analyzing citation's patterns among scholarly articles research tendencies and influences may be discovered.

Intuitively much as citations represent the conferral of authority, but not every citation implies the same authority conferral. For this reason, simply measuring the quality of an article by the number of incoming citations from other papers in not robust enough. A better model should be one in which authority of a document doc_i depends on the number of incoming citations and on the authority of the documents $\{\text{doc}_j\}$ which cite doc_i.

Random walk theory has been widely used to compute the authority of a document emerging only from the topological structure of the citation graph [Cha03, MRS08]. The underlying assumption is that the authority of a document doc_i is modelled by the probability $\Pr(p)$ of reaching that document during a random walk on the graph defined by the link structure.

Summary First in section 3.4.1 we provide a briefly mathematical background on finite-state, discrete-time Markov chains. Section 3.4.2 discusses the necessary

modifications to be applied to a more general graph, as a citation graph, in order to leverage Markov theory. Section 3.4.3 uses previous considerations to perform the link analysis over our test collection, finally in section 3.4.4 we point out concerns about taking into account link analysis in the shaping of the notion of user's needs.

3.4.1 Finite-state, discrete-time Markov chains

A finite-state discrete-time Markov chain is a stochastic process which occurs in a series of time steps in each of which a random choice S_i is made. The possible values of S_i form a countable set S called the state space of the chain. Markov chains are often described by a directed graph, where each edge p_{ij} is labeled with the probability of going from state S_i to state S_j.

Stochastic matrix p_{ij} is the probability that the state at next step will be S_j, conditioned on the current state being S_i. Each entry p_{ij} is known as a transition probability and depends only on the current state S_i (Markov property). Since at any given time step k the Markov chain can be in one of the n states, entries in each row add up to 1 (cf. equation 3.12b).

$$\mathbf{P} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \quad p_{ij} \in [0,1] \tag{3.12a}$$

$$\sum_{j=1}^{N} p_{ij} = 1 \tag{3.12b}$$

$$p_{ij} = \Pr(X_{k+1} = S_i | X_k = S_j) \tag{3.12c}$$

Periodic Markov chain A state S_i is *periodic* if the probability of returning to the state S_i is zero except at regular intervals. Formally, the period of a state is defined as the greatest common divisor of the set of possible returning intervals in the considered state S_i:

$$k = \gcd\{n : \Pr(X_n = S_i | X_0 = S_i) > 0\} \tag{3.13}$$

If $k = 1$ the state S_i is said to be *aperiodic*, otherwise (i.e. $k > 1$) the state is said to be *periodic* with period k.

Irreducible Markov chain A Markov chain is said to be irreducible if its state space is in a single communicating class i.e. if there is a non-zero probability of transitioning (even if in more than one step) from any state to any other state.

Mathematically, the corresponding probability matrix $\mathbf{P}$ is said to be irreducible if there exists a permutation matrix $\mathbf{Q}$ such that:

$$\mathbf{QPQ}^T = \begin{pmatrix} \mathbf{A}_1 & \mathbf{B} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix} \tag{3.14}$$

Ergodic Markov chain A state S_i is said to be *ergodic* if it is *aperiodic* and *positive recurrent*[1]. If all states in a Markov chain are ergodic, then the chain is said to be ergodic. It can be shown that a finite-state irreducible Markov chain is *ergodic* if its states are *aperiodic*.

Steady-state analysis We can depict the probability distribution of a random walk over the states at any time step k by using a probabilty vector[2] $\mathbf{x}$. At step $k = 0$ we can start the walk may begin at a state S_i whose corresponding component in the vector $\mathbf{x_0}$ is 1. After k time steps the probability to be in each state of the chain is given by equation (3.15)

$$\mathbf{x}_{k+1} = \mathbf{x}_k\mathbf{P} \tag{3.15}$$

If we continue a random walk over the Markov chain, each state is visited at a different frequency which depends only on the topological structure of the chain.

Theorem For any *ergodic* Markov chain, there is a *unique* steady-state probability vector $\mathbf{x}_s$ which is the principal left eigenvector of $\mathbf{P}$, such that if $\eta_{i,k}$ is the number of visits to state S_i in k steps, then:

$$\lim_{k\to\infty} \frac{\eta_{i,k}}{k} = x_i \tag{3.16}$$

If $\mathbf{x}_0$ is the initial distribution over the states of the Markov chain, then the distribution at step k is given by $\mathbf{x}_k = \mathbf{x}_0\mathbf{P}^k$, i.e. by recurring applying equation (3.15). As k grows large it is expected that the distribution probability $\mathbf{x}$ attains to a stationary distribution $\mathbf{x}_s$:

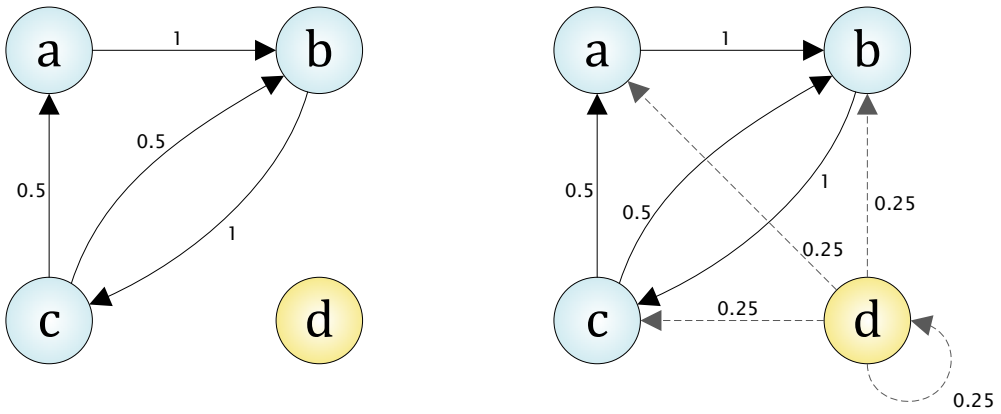
$$\lim_{k\to\infty} (\mathbf{x}_0\mathbf{P}^{k+1} - \mathbf{x}_0\mathbf{P}^k) = \mathbf{0} \tag{3.17}$$

Since the theorem ensures that such distribution $\mathbf{x}_s$ is unique, it does not depend on the initial distribution $\mathbf{x}_0$. The steady-state distribution $\mathbf{x}_s$ can be found by solving the eigenvector problem[3] (3.18), in which the normalization equation

[1] We do not provide the formal definition of recurrence. Intuitively a state is positive recurrent if the expectation of returning in that state it is finite, i.e. the state is not transient.

[2] Note that here we treat $\mathbf{x}$ as a row vector

[3] In this formulation, in which $\mathbf{x}$ is a row vector, the $\mathbf{x_s}$ which satisfies the problem 3.18 is known as the left eigenvector of $\mathbf{P}$



(a) d is a dangling node since it has no outgoing edges. Therefore the transition probability matrix which describes the chain is not stochastic. Note that the sub-graph $\{a, b, c\}$ is irreducible (i.e. each node is reachable from any other node) and aperiodic, therefore it is ergodic.

(b) In order to make the transition probability stochastic we need to add to each dangling node some outgoing edges, e.g. one edge for each state. Note that the graph is reducible since the state d is not reachable from any other node, i.e. the graph is made up of two communicating classes. Therefore the chain is not ergodic.

FIGURE 3.6: Required modification to a generic graph, which comprises dangling node, in order to make it ergodic. This guarantees the existence of a unique steady-state probability distribution to being in each state after a random walk over the states.

ensures that $\mathbf{x}_s$ is a probability vector.

$$\begin{cases} \mathbf{xP} = \lambda \mathbf{x} \\ \sum_j x_j = 1 \end{cases} \tag{3.18}$$

3.4.2 Application to link analysis

Consider the small graph in figure 3.6(a) whose topology is summarized by adjacency matrix 3.19

$$\mathbf{A} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{3.19}$$

We assume for ease that the probability to moving from a state S_i to one of its direct successors is the same for all of them, i.e. if state S_i has k direct successors, the probability to move from S_i to one of them in one step is $1/k$. The probability

to move among states is summarized in the probability matrix 3.20:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.20}$$

Note that the transition probability matrix (3.20) is not stochastic (cf. equation 3.12) since there is a row which contains all zeros. This occurs whenever the graph contains *dangling nodes*, i.e. nodes which contains no outgoing edges ad node d in figure 3.6(a). In order to make (3.20) stochastic we may modify the graph by adding to each dangling node an edge to each of the n nodes (including itself) in the graph, each of them with associated transition probability of $1/n$. By allowing this expedient we yield the stochastic matrix (3.21):

$$\mathbf{P_s} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} \tag{3.21}$$

However, in order to make use of the steady state analysis it is required an ergodic Markov chain i.e. one that is both irreducible and aperiodic. Note that in our example the chain depicted in figure 3.6(b) is reducible since there is no path to return to state d, i.e. there are two communicating class $\{a, b, c\}, \{d\}$.

Perturbation of the chain A common adopted solution to alter a graph with no dangling nodes (i.e. whose transition probability matrix is stochastic, irreducible and aperiodic therefore ergodic) is to consider a convex combination of the stochastic matrix $\mathbf{P}$, which has no guarantees on reducibility or periodicity (as the one in our example), with a stochastic perturbation matrix[4] $\mathbf{E}$ [PBMW98, LM04, LM05, LM06, BGS05].

$$\mathbf{P_e} = \alpha \mathbf{P}_S + (1 - \alpha)\mathbf{E}, \ \alpha \in [0, 1] \tag{3.22}$$

Google PageRank [PBMW98] gives back a practical reason to that perturbation matrix $\mathbf{E}$ by modelling the behaviour of a random surfer which randomly choose to jump to a new Web page by manually entering an URL, i.e. without following outgoing links.

[4] A stochastic perturbation matrix models a chain in which each node has an edges to each other node (comprising itself) in the graph, i.e. there is a probability to go from each state to each other state in one step

Going back to our example, by choosing $\alpha = 0.8$ we yield the (3.23) which models an ergodic Markov chain:

$$\mathbf{P_e} = \begin{pmatrix} \frac{1}{20} & \frac{17}{20} & \frac{1}{20} & \frac{1}{20} \\ \frac{1}{20} & \frac{1}{20} & \frac{17}{20} & \frac{1}{20} \\ \frac{9}{20} & \frac{9}{20} & \frac{1}{20} & \frac{1}{20} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \tag{3.23}$$

Since the Markov chain described in (3.23) is ergodic, the theorem guarantees that it has a unique steady-state distribution of probabilities to be in each of its state during an arbitrary long random walk over its states. By solving the corresponding eigenvector problem as stated in equation (3.18) we yield (3.24) which reveals that the probability of visit the state b is higher since it is easily reachable from other nodes (cf. figure 3.6).

$$\mathbf{x_s} = \begin{pmatrix} 0.206 & 0.371 & 0.360 & 0.062 \end{pmatrix} \tag{3.24}$$

3.4.3 Implementation

Each scholarly document is represented as a node in a very large graph. Directed arcs connecting these nodes represents citations between documents.

For our experiment we used a freely available Python implementation[5] of PageRank algorithm. It relies on power iteration algorithm in order to compute the steady-state distribution $\mathbf{x_s}$, i.e. it iterative applies the equation (3.15) unless the difference (3.17) between two subsequent steps becomes less than a specified amount ϵ. The applicability of the power iteration method is possible since the ergodic property ensures that the convergence is fast. We performed the link analysis over the same data set discussed in figure 2.11. As we can see in figure 3.7 the link analysis highlights the most authoritative items within the collection.

3.4.4 Applicability and criticism

However the use of the link analysis in order to provide a profile of the user's preference and his research directions needs to be carefully evaluated. Authoritative items are biased to older items so the research directions are not proper highlighted. However, the link analysis can be considered as one of the possible signals which comes from an analysis of the collection features that need to be evaluated by a tool which tries to combine the desired features from those signals.

[5] http://kraeutler.net/vincent/essays/google_page_rank_in_python

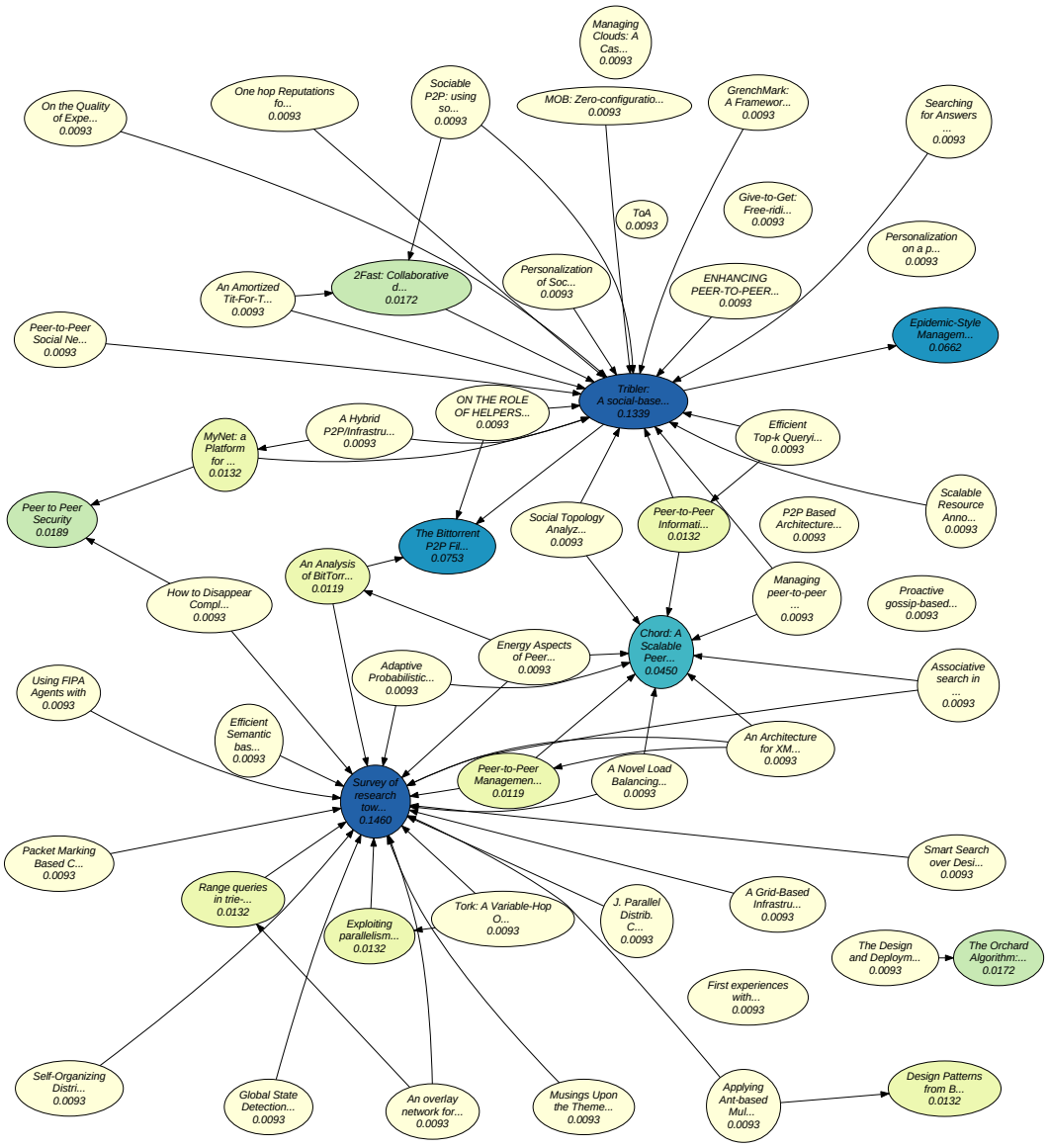


Figure 3.7: Probability to visit documents during a random walk over the directed graph induced by references. The result was computed by making use of PageRank algorithm with a perturbation coefficient $\alpha = 0.85$ (cf. equation 3.22). Note that the authority of a documents depends on the authority of the citing document. As example link analysis could be used to discover user's preferences and research trends. However link analysis has a bias towards older papers which are more likely to be cited.

Chapter 4

Distributed content location

Information retrieval system task lies in retrieving the information which is expected to satisfy user's information needs. This is carried out by picking out of the entire collection those documents which appears to be relevant according to a certain relevance model. Such a system, which requires a global knowledge of the information, leads on the adoption of a client-server paradigm.

The main difficult while relying upon peer-to-peer computation model lies in the cost required by maintaining a global knowledge of the network from each peer. Since this cost becomes unacceptable as increasing network size, recent researches have proposed to map the semantic proximity of information into the network connection topology which should be exploited while locating content. Assuming the cluster hypothesis, network topology can be proactively modified in order to push each node towards relevant information.

Section 4.1 discusses about concerns on locating contents and points out problems while claiming for a fully decentralized approach. Section 4.2 outlines models which have been employed while distributing an information retrieval system and discusses about limitations emerging while dealing with large scale highly-dynamics networks. Section 4.3 discusses about Semantic Overlay Networks which leads to the topology construction problem. Section 4.4 reviews a gossip-based approach for proactive topology management.

4.1 Problem definition

The task demanded to an Information Retrieval (IR) system is to provide access to information which may be distributed over different locations. A user in need of some information issues a query to the IR system which picks out of the entire collection those documents which are expected to be relevant with respect to the user's information need. The selection of relevant documents is performed by resorting on a relevance model which has a global access to information, i.e. it knows about its existence.

4.1.1 Client-server mapping

A client-server computing paradigm fits better this requirements of having a global view of the information. In case of contents are spread out the entire network it is necessary to let the service know about its existence, i.e. contents need to be published on the service in order to be located afterwards.

Publishing contents Publishing can be done by using a pull or a push approach. The former paradigm, which is employed by Google's spiders [BP98], resorts on the use of agents which traverse the network looking for new contents. The latter follow a symmetrically approach for which information sources are in charge of publish contents by contacting the IR-service. This approach was adopted by the Napster peer-to-peer file sharing service in which each peer wishing to share some contents contacted a central service in charge of keeping track of the whole shared contents. BitTorrent content distribution protocol [Coh03] relies on Web server in which information about contents (i.e. `.torrent` files) are published in order to be distributed.

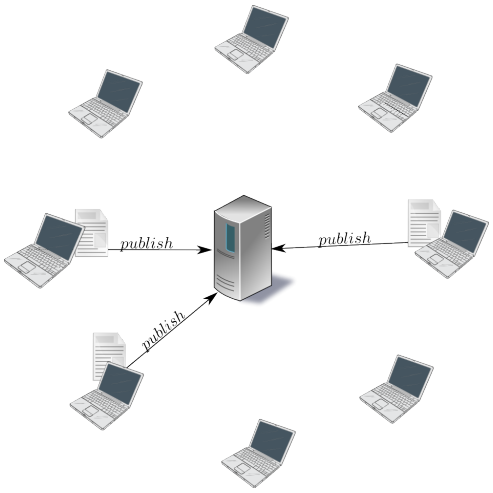
Criticism A user wishing to search for some information plays the role of the client which issues a query to the central server. As for every computation which relies on a client/server distribution model, this suffers for scalability problems both in increasing number of query requests and size of indexed content. A general solution to these limitations relies on replication and distribution. Google adopts a strategy for which individual tasks of an information retrieval service, (e.g. document indexing, document analysis, retrieval task) are demanded to cluster of servers which are highly coupled.

Clearly this approach requires large investments in resources as increasing service requirements. Moreover a solution based on a client-server approach has high dependence on predefined central server which results in possible exploitation of private data.

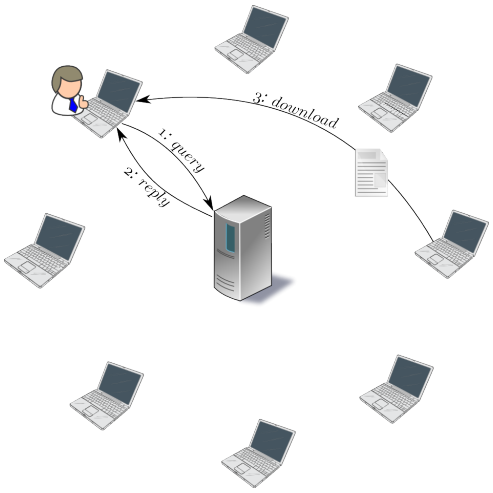
4.1.2 Peer-to-peer mapping

In peer-to-peer (P2P) computational paradigm each node plays the same role (i.e. it is a *peer*) meaning that the computation is fully distributed. Since there is not privileged node over the entire population, the main difficult of locating content within such a scenario lies in the cost required to allow each node to have a global knowledge of the entire network. This can be done by accepting high levels of overhead which are required for acquiring a global view of the network.

Different solutions have been proposed [ATS04, RM06] for distributing IR-systems over P2P networks. There is no absolute best choice among them [ZMSM05], but their adoption depends on networks size and dynamics, number



(a) In order that content can be searched, central service must know about it. A push approach requires that clients publish their content to the central service (e.g. BitTorrent). In contrast a pull paradigm requires an agent which traverses the network looking for new contents (e.g. Google's spiders).



(b) A client, which needs some information, issues a query to the central service which replies providing the location in which the desired information resides. Then the client contacts the target location asking for the required information.

FIGURE 4.1: Protocol employed while interacting with a centralized information retrieval system which has a global knowledge of the content which is spred out the entire network. Information retrieval systems such Google follow this paradigm. Content location services of Napster and BitTorrent file sharing systems rely also on this approach.

of documents, query throughput. They could be placed into different taxonomies depending on the extent of the distribution, i.e. from the heterogeneity of client-server architectures to the complete homogeneity of pure P2P systems, or on the kind of communication topology whit which nodes are connected each other.

Requirements We are interested in a fully decentralized keyword-based IR service which can be used for large scale highly-dynamics P2P network such as P2P content distribution networks.

4.2 Distribution models of keyword-based IR systems

In this section we outline the costs of naive implementations of two straightforward means of distributing a keyword-based IR-system in which the user's query is a set of keywords and the indexed documents contains text. The basic tool which have been widely used for this purpose is the Vector Space Model (VSM) which turns on a vector space both documents and queries in order to ease the

computation of relevance (cf. section 3.1). The knowledge of the whole collection is encapsulated within a very large term-by-document matrix (equation 3.1). This large matrix can also be viewed as an index containing a list of terms, for each of them a posting list stores the list of documents in which that word occurs (cf. figure 3.1). These posting lists are intersected while a query involves more than one keyword.

An IR system has a global knowledge of the content while it has access to that very large term-by-document matrix. Two straightforward proposals could be employed in order to distribute these large matrix over multiples nodes.

4.2.1 Partition-by-document

Partition-by-document scheme splits the term-by-document matrix by columns. Documents are divided up among peers each of them maintains a local inverted index of the documents it is responsible for. Each query posed by a user must be broadcast to all peers of the systems, regardless to the number of keywords in the query. Each of them returns its most highly ranked documents to the requester node which will merge all the results by ranking them in a unique list.

4.2.2 Partition-by-term

Partition-by-terms scheme splits the term-by-document matrix by rows. Responsibility for the words which appears in the document corpus is divided up among the peer population. Each peer stores the posting list of the words it is responsible for.

A query involving multiple terms requires that posting list concerning one or more terms be sent over the network. This scheme ensures that, in order to serve a query containing n keywords, no more than n nodes are involved in the coordination, which requires intersection of posting lists located on different nodes. Communication cost grows linearly with the number of documents in the system.

4.2.3 Support of DHTs middlewares

Different policies can be employed while partition the term-by-document matrix according to one or a combination of the two strategies described above. If an exact knowledge of the content is required (as a database view) we should provide a unique name to each item, typically by using an URN scheme.

While partition the term-by-document matrix each item (term, or document) should be mapped into a node by relying on middlewares which provides Distributed Hashing Tables (DHTs) service. They provide distributed hashing services by hiding names to the physical object location. These middlewares can also ensures availability and fault-tolerance by relying on replication of objects, obviously by accepting an extra overhead.

Applications database-like in which exact recall are needed should typically be built by using those partition schemes which relyies on a DHT middleware service.

4.2.4 Literature proposed solutions and scope of usage

A number of IR-system built on top of DHTs middlewares have been proposed. [RV03, LLH+03] evaluate the cost required for those applications and mapped out some possible optimizations such as caching, pre-computation and compression in order to limit the required bandwidth. However the cost required for deploying this kind of system are infeasible for large scale applications such as P2P file sharing systems [YDRC06, ZMSM05]. Moreover while dealing with high node churn and unavailability which typically affects large scale high-dynamics P2P DHT-based approach requires an overhead is not documented to date for those kind of networks.

Scope For these reasons this kind of applications should be employed in those situation in which high overhead can be accepted, e.g. services with an underlying solid business model, federation of digital libraries which have resources and in general in those scenario in which there is no high dynamics regarding node population and document publishing.

4.2.5 Query routing over random connection topologies

By give up database oriented application in which there are constraint on object naming, some forms of object replication should be accepted. Object replication is due to its popularity over the nodes. In such scenario partition-by-document scheme have been employed.

A query issued by a peer should be broadcast to all peers in the population in order to have guarantees of recall (cf. figure 4.2).

Gnutella's flooding Gnutella uses broadcast based on flooding in order to locate content. When a node issues a query it send the request to its neighbours which in turn propagate the query to their neighbours and so on until the query reaches all the clients within a certain radius from the original querier. In order to limit the number of hops each query has limit to maximum number of hops in which it can be forwarded. By adopting this solution each peer establish a virtual horizon beyond which their messages could not be forwarded.

This solution can easily locate very popular items which are replicated in many nodes out of the entire population. Items in file sharing systems are distributed according to the Zipf's law, i.e. few items are highly replicated across node population while other items are extremely rare.

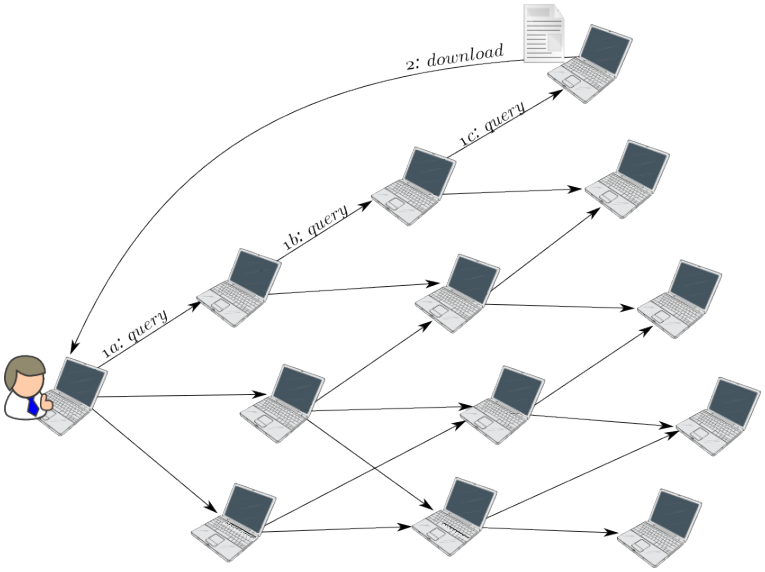


Figure 4.2: Protocol employed while locating content on a network in which each peer knows only about its content. A peer wishing to retrieve some content must broadcast the query to the whole network in order to acquire a global knowledge of the content location. A common solution adopted for query broadcast in case each node has only a partial view of the entire network resorts on flooding (e.g. Gnutella). Main drawback of this approach is huge overhead required for broadcast whit consequent scalability limitations.

Because load on each node grows linearly with the total number of queries, which in turn grows with system size, this approach does not scale for large systems.

Exploiting connection topology

Connection topology of Gnutella's peers are completely random, i.e. unstructured. A query is simple broadcast over the network by using this random topology. In order to improve scalability random walks have been proposed [LCC+02] to replace flooding. A query is forwarded to a randomly chosen neighbour at each step until sufficient responses to the query are found. However, as flooding, random walks are essentially a blind search since at each step a query is forwarded to a random node without taking into account any indication of how likely it is that node to have responses for the query.

Several modification to Gnutella's design have been proposed in order to improve scalability. Connection topology should be modified in order to avoid the traffic which comes from flooding. For instance node heterogeneity could be exploited in order to adapt the overall connection topology avoiding overloaded hot spots [CRB+03]. PlanetP [CAPMN03] summarizes contents on each node and

floods the summaries to the entire system in order to forward the query only to a small subset of promising peers. [CFK07] uses guide rules to organize nodes into an associative network.

4.3 Semantic Overlay Networks

Gnutella scalability problems are due to the random connection topology of the peers. Since there is no global knowledge of the content spread among nodes a query is forwarded with a blind mechanism towards neighbours nodes. Connection topology should be modified in order to organize nodes with similar semantics close to each other. In such scenario broadcast could be avoided by routing the query towards those nodes which are expected to be closer to the query semantics. That kind of network topologies, which aggregates node with similar contents, are known as Semantic Overlay Networks (SONs). In these network topologies contents are organized around their semantics such that the distance (e.g. routing hops) between two documents in the network is proportional to their dissimilarity in contents.

When a nodes issues a query a routing mechanism should route the query towards the region which are close in semantics to the query.

Solutions adopted in literature pSearch [TXD03] maps each document into points of a semantic vector space. Each document is placed into a node by resorting on the use of a Content Addressable Network (CAN) which provides DHTs service. When a peer issues a query, this is projected into a semantic subspace and based on this it is routed towards the peers which are close in semantics as a classic nearest neighbour search strategy. As this solution appears to be efficient and elegant it should be noted that pSearch relies on the use of some global statistics as the basis of the semantic space which must be distributed over all the nodes in the network.

4.3.1 Strategies on exploiting network topology

There are two strategies on exploiting network topologies for improve the content searching.

Reactive approach Reactive approach makes usage of the underlying search mechanism in order to infer semantic relations based on the query placed and the corresponding replies received. The network topology is changed each time a user poses a query to the system [VKMvS04, SMZ03]. They generally rely on heuristics to decide which peers that served a node recently are likely to be useful again for future queries. One of the hidden assumptions those solutions make is that the network is static both in terms of node joining and leaving and changes

in nodes' profiles. [VKMvS04] examines different policies in order to modify the network topology to improve search performance.

Proactive approach Proactive approach builds a semantic overlay topology by making use of a peer profile and a ranking function which induces an order over the peers. The network topology construction is done proactively in a completely implicit way, that is decoupled to the execution of queries. The peer profile could capture the user's search trends through analysis of query posed, the results and his file cache.

The reactive approach minimize the resource usage, but leads to low hit ratio. Proactive approach his more intrusive but it can leads in high quality semantic overlay topologies, based on the choice of a proper ranking function and a peer profile.

Locality of interest principle

Locality of interest principle posits that if a peer has a particular piece of content that one is interested in, it is very likely that it will have other items that one is interested in as well. This has been demonstrated by examining traces of content distribution applications such as P2P file sharing systems [SMZ03].

4.3.2 Topology construction problem

We consider a set of processes distributed over a network. Each process is reachable by using a network address and a port, which is necessary and sufficient for sending it a message. Each node knows about c other nodes of the networks through a partial view which is a set of node descriptors. The local knowledge of each node about the network, i.e. their partial views, define the links of the overlay network topology. In addition each node p_i has a profile π_i as part of its descriptor. The input of the problem is a set of n nodes, the view size c and a ranking function ρ defined over the node profiles Π which induces an order over the nodes $p_1, \ldots, p_n$.

$$\rho : \Pi \times \Pi \mapsto \mathbb{R} \tag{4.1}$$

The goal is to build the views of the nodes such that for each node p_i its view $v(p_i)$ contains exactly the first c elements according to the order induced by the scoring function ρ out of the whole network.

Ranking functions The topology of the network is defined by the choice of the ranking function ρ which induces an order over the nodes. It can be based on an approximate property of components including geographical location, semantic proximity, available bandwidth or simply on an abstract, pre-defined topology over an ID-space like a ring or a torus. Since we are interested in clustering nodes

which are close in their semantics we should resort on a ranking function defined over the node's profile which takes into account semantic proximity.

One way of generating ranking functions is through the distance function $\rho(\pi_i, \pi_j)$ where the profiles π_i, π_j of the nodes are elements in a metric space. In such case the ranking function can simply order the given set according to the distance between elements.

A metric space is such that for any element in the set it holds the triangle inequality:

$$\rho(\pi_x, \pi_z) \leq \rho(\pi_x, \pi_y) + \rho(\pi_y, \pi_z) \tag{4.2}$$

4.4 Gossip-based topology management

Topology management can be identified as an abstract membership service middleware which eases development of applications by hiding distributing concerns. There has been some proposal for creating and managing a network topology in a large scale, dynamic, fully distributed system. Because these settings all the proposed solution are based on a gossip-based probabilistic solution to this problem.

Section 4.4.1 gives an overview of gossip protocols and highlight scope of usage. We focus on solution which use a proactive approach, i.e. autonomously build a topology my making usage of the peer's profile and ranking function as stated in the topology construction problem in section 4.3.2. Section 4.4.2 delineates desirable properties of ranking function and node's profile while dealing in those settings. Section 4.4.3 discusses on peer sampling service which allows a probabilistic global view of the network to each peer. Section 4.4.4 reviews two solutions which have been recently proposed for proactive management of topologies which leads to the construction of a semantic overlay network.

4.4.1 Gossip protocols

Gossip protocols (also known as epidemic protocols) are probabilistic multicast schemes in which no hard guarantees are given concerning the delivery of a multicast message. They are a model to spread a piece of information among a large number of processes with a dynamic connection topology. They exhibit a very robust and scalable features even in the presence of a high rate of link failures. They have been studied theoretically and their analysis is built on mathematical foundations. The survey [EGKM04] provides an introduction to the field and delineates some key issues while designing a gossip protocols.

Typical application The use of epidemics algorithms has been explored in applications such as database replication, failure detection, data aggregation, resource discovering and monitoring. We will make use of epidemic protocols in order to

dynamically build a network topology which reflects the semantic clustering of information.

Model The principle underlying this information dissemination techniques mimics the spread of epidemics. Another close analogy is with the spread of a rumour among humans via gossiping.

A process that wishes to disseminate a new piece of information to the population, does not need to know all the participants, which membership could have high dynamics, but only a small subset of other peer processes.

Every process buffers every message (information unit) it receives up to a certain buffer capacity b. It forwards that message a limited number t of times to a randomly chosen subset of processes of limited size f (i.e. the fanout of the dissemination). The reliability of information delivery will depend both on these values as well as on the size n of the whole population.

4.4.2 General requirements

We assume dynamic collection of distributed nodes that want to participate in a common epidemic protocol. Nodes may join and leave and they may crash at any time resulting in a possible high node churn rate. There are two sides to the construction of a topology under these settings. It is desirable to chose a ranking function which holds the triangle inequality (4.3c) since it eases the use of epidemic algorithms while managing topologies. Exploiting the triangle inequality property of the ranking function (4.1) should quickly leads to high quality local views. For instance if node q is in the local view of p and r is in the local view of q it makes sense to check whether r is also close to p.

Node that triangle inequality property of the ranking function does not constitute an hard requirement for the system. In its absence closer neighbours are discovered based only on random encounters.

Candidates from all over the networks should be examined. The problem with examining only neighbours' neighbours is that we will be eventually searching only within a single cluster, although a good neighbour may have just joined at a random point in the network. Likewise, when new nodes joins the network or they quickly changes their profile they should easily find an appropriate cluster to join. These issues call for a randomization when selecting nodes to inspect for adding to a local view.

4.4.3 Peer sampling service

Epidemic protocols makes the assumptions that each node has a full view of the network since each node periodically gossips with a random node out of the whole

set. In reality, scalability concerns requires that each node has only a partial view of the network, i.e. it knows just a set of c neighbours.

Peer sampling service aims to provide to each node a probabilistic global access to the entire network. As the input it takes the distributed collection of nodes and as output it returns a peer as a result of an independent uniform random sampling among the collection.

Newscast Newscast is a highly reliable and fully distributed dissemination scheme for implementing information dissemination and membership management [JKvS03]. It leads to a small world network, i.e. a topology connection with features as high clustering and low diameter. These features are bad for flooding because it results in many redundant messages and in self-healing since strongly connected clusters are weakly connected to the rest of the network.

Cyclon Cyclon is a gossip-based membership management protocol which leads to the construction of graphs which have low diameter, low clustering and highly symmetric node degrees which results roughly in a random graph [VGVS05]. Moreover it is highly resilient to massive node failures by reactive restoring randomness. It offers a fully decentralized service for delivering information of new events that can happens everywhere in the network.

Importantly it can achieve this property fairly quickly even when a small number of items (such 3 or 4) is exchanged in each communication. Therefore it is ideal as a lightweight service that can offer a node randomly selected peer from the current set of nodes.

4.4.4 Topology management protocols

T-Man [JB06] gradually evolves the topology to match it to one defined by the ranking function by using only local gossip messages. However it does not takes into account the network dynamics concerning both nodes churn and variation in peer's profile or in the ranking functions.

In order to adapt to network dynamics a topology management protocol should sample the network for changes. These requirements calls for the use of a peer sampling service which gives a support to the dynamics by a probabilistic access to the network.

A gossip-based topology management protocol which dynamics requirements should resorts on a two-layered set of gossip protocols in which a peer sampling service, which lies in the lower layer, provide a probabilistic access to the whole network to the top protocol which is responsible for manage the overlay network topology.

This approach have been proposed for gossip-based management of semantic overlay networks [VvS05, VvSI07]. The top layer protocol *Vicinity* is in charge

Algorithm 5 Cyclon and Vicinity epidepic protocols skeleton

```
1: procedure ActiveThread                    ▷ Runs periodically every T time units
2:     q ← selectPeer()
3:     myItem ← (myAddress, timestamp, myProfile)
4:     bufSend ← selectItemsToSend()
5:     send bufSend to q
6:     receive bufRecv from q
7:     view ← selectItemsToKeep()
8: end procedure

9: procedure PassiveThread                   ▷ Runs when contacted by some peer p
10:    receive bufRecv from p
11:    myItem ← (myAddress, timestamp, myProfile)
12:    bufSend ← selectItemsToSend()
13:    send bufSend to p
14:    view ← selectItemsToKeep()
15: end procedure
```

discovering peers which are semantically as close as possible and on adding these nodes to the local view of each peer. It resorts on the use of Cyclon peer-sampling service. It has been showed through simulations that this multi-layer approach ensures rapid convergence of network topology and at the same time guarantees highly adaptivity and low overhead.

Tribler BuddyCast These ideas have been applied to the design of BuddyCast [PYM+08], i.e. the first large-scale internet deployed epidemic protocol stack for managing dynamically evolving network topologies. The main features are discovery of peers, discovery of content and formation on semantic overlay. All of these services are performed in a complete decentralized fashion. BuddyCast forms the core of the peer-to-peer file sharing system Tribler [PGW+08].

Design key issues

In this section we point out key concern while design a gossip-based overlay network management protocol. We refer to the solution proposed in [VvS05, VvSI07].

Gossip items All information exchange between peers is carried out by means of gossip items. A gossip item created by peer p_i is a tuple containing the following three fields:

- peer p_i's contact information (e.g. network address and port);
- item's creation time, i.e. timestamp;

- peer's profile;

Each node maintains locally a number of items per protocol called the protocol's view.

Cache size A large cache size provides higher chances of making better items selection and therefore accelerate the construction of near-optimal semantic views. On the other hand the larger the cache size, the longer it takes to contact all peers in it, resulting in the existence of older and therefore more likely to be invalid links. Experiments have been conducted with a cache size of 50 items for each of the two layers.

Gossip length The number of items exchanged in each communication is predefined and is called the protocol gossip length. It is the number of items gossiped per gossip exchange per protocol and it is a crucial factor for the amount of bandwidth used. It is g_V for Vicinity and g_C for Cyclon. Even though exchanging more items per gossip exchange allows information to disseminate faster, low-intrusion concerns calls for a trade-off. Experiments have been conducted by keeping the gossip length for each of the two layer equal to 3.

Gossip period The gossip period is a parameter that does not affect the protocol behaviour. The protocol evolves as a function of the number of messages exchanged. The gossip period only affects how fast the protocol's evolution will take place in time. The single constraint is that the gossip period should be adequately longer than the worse latency throughout the network, so that the gossip exchanges are not favoured or hindered due to latency heterogeneity. A typical gossip period would be 1 minute.

Node profile and ranking function Construction of the topology relies on the definition of an order over node's. While dealing with gossip protocols convergence and adaptivity to dynamics are better achieved by resorting on a triangle inequality (4.3c).

A node profile $\pi_i \in \Pi$ should be elements of a metric space (Π, ρ) in which Π is the set of node's profile and the ranking function ρ is a metric on Π, i.e. a distance function which holds the following conditions:

$$\rho(\pi_x, \pi_y) = 0 \Leftrightarrow \pi_x = \pi_y \tag{4.3a}$$

$$\rho(\pi_x, \pi_y) = \rho(\pi_y, \pi_x) \tag{4.3b}$$

$$\rho(\pi_x, \pi_z) \leq \rho(\pi_x, \pi_y) + \rho(\pi_y, \pi_z) \tag{4.3c}$$

Bandwidth considerations Due to the periodic behaviour of gossiping, the price of having rapidly converging protocols leads to high usage of network resources (i.e. bandwidth). In each cycle, a node gossips on average twice (i.e. once as an initiator and once as a responder). In each gossip $2 \cdot (g_V + g_C)$ items are transferred to and from the node, resulting in a total traffic of $4 \cdot (g_V + g_C)$ items for a node per cycle. Since an item's size is dominated by the profile it carries, by assuming that a node's profile takes on average ϕ bytes we approximate with this the item's size. So in each cycle the total number of bytes transferred to and from the node is $4 \cdot \phi \cdot (g_V + g_C)$. For $g_V = g_C = 3$ the average amount of data transferred to and from a node in one cycle is $24 \cdot \phi$ bytes, while for $g_V = g_C = 1$ it is just $8 \cdot \phi$ bytes. With $g_V = g_C = 3$ the systems adapts a little faster to changes, but if bandwidth is of high concern (minimal intrusion) $g_V = g_C = 1$ can also provide good results.

Chapter 5

A proof-of-concepts prototype

This chapter makes usage of the insights acquired in previous chapters in order to design a proof-of-concept prototype of a fully distributed text-based search and recommendation system. Individual task of the information system are described highlighting the employed solution, which represents the fundamental of the system, and future extensions towards the state-of-the-art.

Section 5.2 describes the subsystem in charge of building a local database of scientific literature which constitute the key feature of the system. Section 5.3 concerns on meaning of learning user's information needs by relying on a certain number of objective measurements which includes the analysis of the locally built database. Section 5.4 concerns the definition of the user's profile and ranking function which are used by a gossip-based topology management in order to set-up a social network of users. Section 5.5 discusses how to exploit the social network in order to locate and recommend interesting content to the user. Section 5.6 describes our prototype which have been used for an emulation of the system.

5.1 System architecture

Overall system architecture and data flow through components are depicted in figure 5.1.

5.2 Collecting data

Papers which have been read by the users are locally indexed. In section 2.4 we discuss the design, the implementation and the effectiveness assessment of a component which automatically extracts metadata, full-text and cited references from PDF files research papers in order to set up a local scientific literature database. The main concern of this task is related to the automatic extraction of structured information from presentation-oriented unstructured text document such as the popular PDF. A state-of-the art solution adopts a machine learning algorithm for

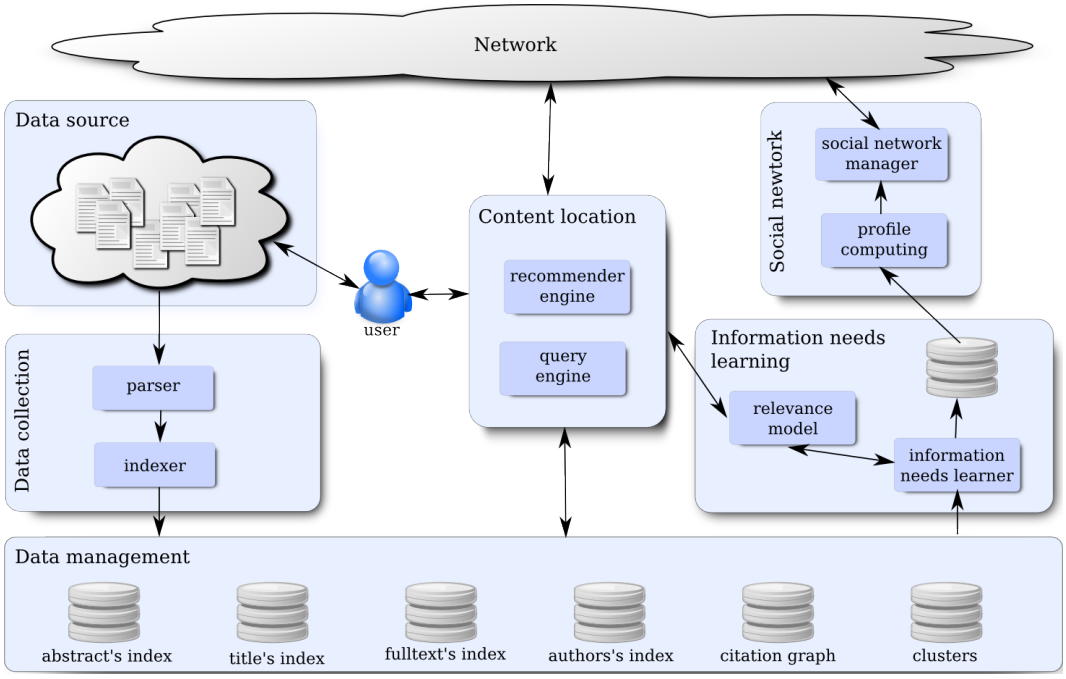


FIGURE 5.1: Data flow through the information system architecture. Metadata, full-text and cited references are automatically extracted from PDF files research papers in order to set up a local scientific literature database which is exploited for automatically learn user's information needs. This are summarized into a profile which is gossiped over the network in order to find user's with similar information needs. Content location and recommendation are carried out by exploiting the dynamically evolving social network.

giving back semantics to text chunks.

5.3 Learning user's information needs

In order to automatically push the user towards the information which he finds as relevant, his information needs should be learned relying upon his interaction behavior with the information system. A machine learning approach should be employed. A number of objective measurements of this behavior could be considered, such as the papers which have been read, the time spent reading each of them, history of search queries and so on.

A number of objective analysis, which have been reviewed in chapter 3, can be performed over this database. The key model while dealing with text documents is the vector space model of text. However, since it is a lexicon-based model, it suffers from its inability to cope with ambiguity of natural language such as synonymy and polysemy of terms. Latent semantic analysis relies on linear algebraic

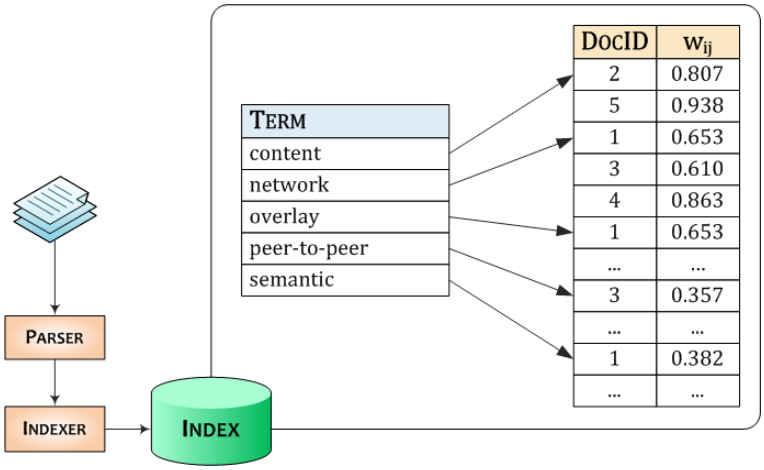


FIGURE 5.2: Text extracted from PDF research paper is used for building an inverted index relying on the vector space model of text.

operations performed over the vector space model in order to extract the text' semantics.

PoC design In our Proof-of-Concept (PoC) design we learn user's information needs by relying just on his local scientific literature database. We considered just the vector space model of text collected from abstract sections since it leads to easier management of the vector space basis which is required in order to compute similarities between user's information needs (cf. section 5.4).

5.4 Social network management

User's learned information needs have to be summarized into a profile which has to be gossiped over the network in order to find users with similar interests. Notion of similarity between user are computed resorting to a certain ranking function.

Gossip-based topology network management advises that the user profile Π with the ranking function ρ to be a metric space (Π, ρ) since by leveraging triangle inequality (4.3c) it leads to faster adaption to network dynamics (cf. section 4.4.2).

5.4.1 User's profile

User's profile should be a lightweight summary of the user's information needs which can be be easily gossiped over the network.

We model users' profile as elements in a vector space in order to ease the computation of similarities between them. This approach requires that elements

are in the same vector space, i.e. each user adopts the same vector space basis. The basis of the vector space model of text is given by the ordered set of terms which have been considered. Latent Semantic Indexing projects document into a lower dimension semantic vector space whose basis is more difficult to manage. For this reason, for our PoC prototype we rely just on the vector space model.

PoC design We considered as a model for user's information needs the centroid $\vec{\mu}$ of the term-by-document matrix as stated in the equation 3.10. In order to keep light the size of the profile $\vec{v}$, we retain just the top-k tf-idf weighted terms of the centroid (5.1a). An evaluation of the approximation extent is given by the norms' ratio (5.1b):

$$\vec{v} = \{\mu_1, \mu_2, \ldots, \mu_k\},\ \mu_i \in (\vec{\mu}, \geq)_k \tag{5.1a}$$

$$\eta = \frac{\|\vec{v}\|}{\|\vec{\mu}\|} \tag{5.1b}$$

The vector profile $\vec{v}$ is then normalized and gossiped through the network in order to find similar profiles by relying on a similarity function ρ defined over this vector space. This approach requires to include within the gossip message the vector space basis, i.e. the list of terms together with their respective weights ν_i.

Since the information retained by the gossiped profile affects the resulting social network topology, it should be carefully evaluated. In our emulation we choose to retain just the most top-30 tf-idf weighted distinct term stems obtained by using the Porter's stemming algorithm which collapses together terms of the vector profile with the same stem [Por97].

5.4.2 Ranking function

Similarities between users' infomation needs are computed by relying on the cosine similarity wich have been used for compute document similarities according to the vector space model (cf. equation 3.4). Similarity between two vector profiles ν_j, ν_k is given by (5.2):

$$sim(\vec{v_j}, \vec{v_k}) = \cos\theta_{j,k} = \frac{\vec{v_j} \cdot \vec{v_k}}{\|\vec{v_j}\|\|\vec{v_k}\|} = \frac{\sum_i \nu_{i,j} \cdot \nu_{i,k}}{\sqrt{\sum_i \nu_{i,j}^2}\sqrt{\sum_i \nu_{i,k}^2}} \tag{5.2}$$

Each peer retain the top-h similar users which have been encountered. The choice of the friendships out-degree h affects the social network topology and therefore the quality of content search and recommendations. It affects also scalability which should be carefully evaluated. In our emulation we choose that each node has at most 5 friends.

5.4.3 Bandwidth requirements estimation

Dynamic management of network topology is achieved by employing the gossip-based protocol reviewed in section 4.4.4. Its bandwidth requirements, which have been pointed out in 4.4.4, depends upon the size of the user's profile which are gossiped through the network.

In this section we gives an estimation of this bandwidth requirements. We assume that each word is made up of an average of 8 characters. By using the ASCII encoding scheme each word request on average 8 bytes. Words' weight can be stored by using 2 bytes resulting on roughly 10 bytes for each (word,term) pair.

Under these assumptions a profile consisting of 30 terms requires on average $\varphi = 10 \cdot 30 = 300$ bytes. In each gossip cycle the total number of bytes transferred to and from the node is $4 \cdot \phi \cdot (g_V + g_C)$ in which node profile ϕ is dominated by user's profile, i.e. $\phi \simeq \varphi$ (cf. 4.4.4).

For $g_V = g_C = 3$ the average amount of data transferred to and from a node in one cycle is $24 \cdot \phi \simeq 7$ kB, while for $g_V = g_C = 1$ it is just $8 \cdot \phi \simeq 2.3$ kB. Considering the gossip period T equal to 1 minute, this translates to an average bandwidth of 120 and 40 bytes/s respectively. Increasing the node profile up to 50 words leads to an average bandwidth requirements of $\simeq 200$ and $\simeq 67$ bytes/s respectively.

5.5 Searching and recommending contents

Content search and recommendation are carried out by exploiting the dynamically-evolving social network. While a user issues a query to the system, query is routed exploiting the network topology expecting that relevant information is located in the user's neighborhood since the network topology has been constructed around user's information needs, i.e. user's has been pushed towards the relevant information locality. Effectiveness of this model resides in the ability of the system to learn user's information needs.

By using the same model, user's neighborhood can be queried with the expected user's information needs leading to provide reading recommendation to the user.

Ranking of relevant information are carried out according to a certain user's relevance model which should be take into account a number of objective measurement. As instance we should consider that citations already acts as reading recommendation (cf. appendix A) and they should be considered while designing a recommendation model.

PoC design In our PoC design we resort just to the vector space model. Neighborhood is queried by using the terms of the user's profile in order to recommend the top-k documents which are expected to be relevant to the user. Intuitively

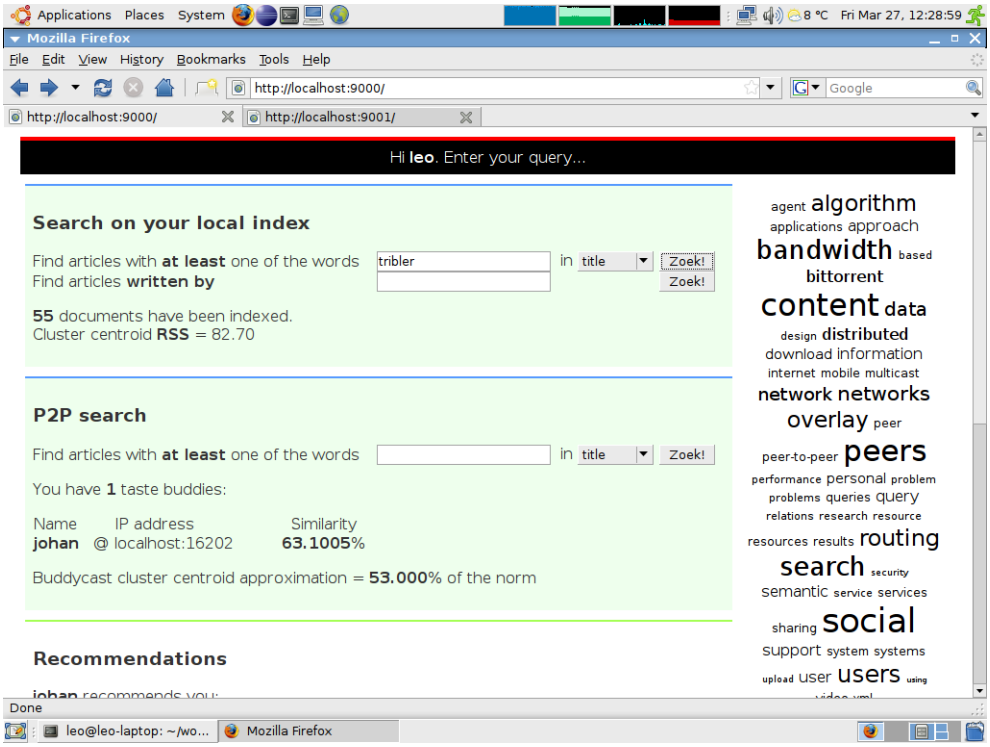


FIGURE 5.3: Screenshot of the system prototype user interface. A web-server has been used for interacting with the system. Term-cloud shows the top-50 weighted terms, according to tf-idf weighting scheme, of the centroid of the term-by-doc matrix built upon the abstract sections of papers. System discovers friendships by gossiping this cloud and using it for calculating cosine similarities. Recommendation of interesting paper are carried out by querying friends with the term-cloud.

each taste buddy recommends his content which are semantically closer to user's preference centroid (e.g. cf. figure 3.4(b)).

5.6 Putting all together: a fully functioning prototype

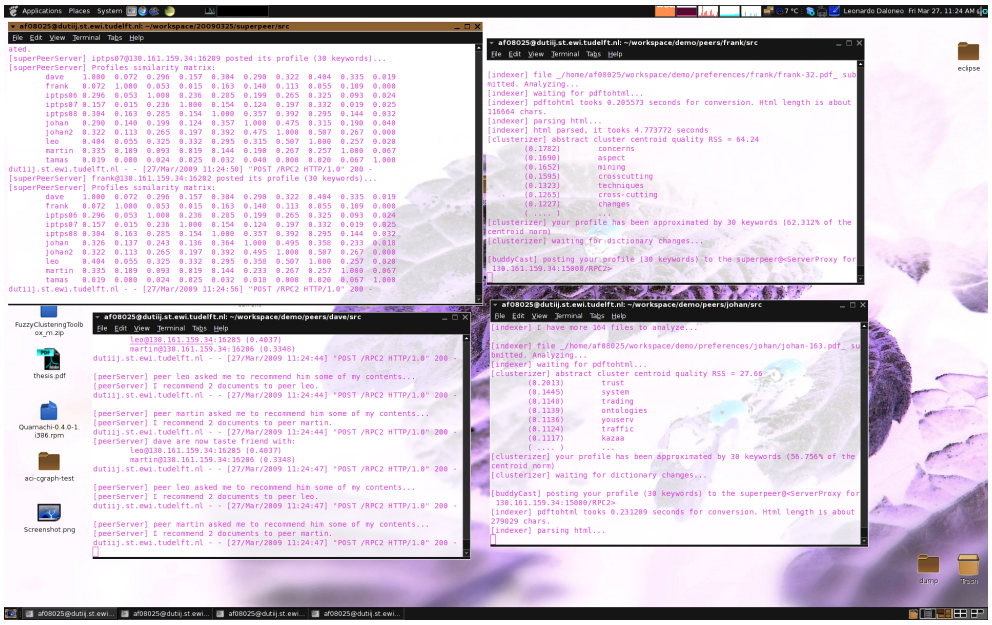


FIGURE 5.4: Screenshot of the standard output of a local emulation of a network made up of just 3 peers and a superpeer in charge of simulating the epidemic-based topology management protocol. Each peer send to the superpeer its vector profile ν. The superpeer computes similarities between peers and builds up the social network topology.

Chapter 6

Conclusions and future work

This chapter gives an overview of the project's contributions. After this overview, we will reflect on the results and draw some conclusions. Finally, some ideas for future work will be discussed.

6.1 Summary and contributions

Content location is a general problem in p2p networks. Most of p2p file sharing systems delegates the task of finding relevant content to a client-server based information retrieval system. This approach requires sizeable capital investments in the infrastructure in order to achieve high scalability levels. Moreover central solutions are prone to possible data exploitation.

6.1.1 Contributions

We designed and implemented a proof-of-concept prototype of a fully decentralized search and recommender system tailored for scientific literature. We proactively maintain a dynamic social network which aggregates users with similar information needs by relying on a lightweight gossip-based topology management protocol. User's information needs are automatically learned by tracking users reading habits. We designed and implemented a prototype of a component which automatically extracts metadata, full-text and cited references from pdf research papers in order to set up a local scientific literature database. This local database is used to learn user's information needs by performing analysis which relies on statistical model of language, such as the vector space model of text. User's learned information needs are summarized into a profile which is gossiped over the network in order to find users with similar information needs. Content location and recommendation are carried out by exploiting the dynamically evolving social network which is expected to satisfy user's information needs. Interesting paper are recommended by users with similar reading interests.

6.2 Discussion and reflection

Our solution is inspired by the fully-decentralized search and recommendation architecture employed in the Tribler file-sharing system [PGW+08]. It dynamically builds a network topology reflecting user's information needs which are automatically learned by tracking user's download preferences. However, a download's history based relevance model is not suitable for text-based content such as scientific literature.

At the same time while we were investigating this solution a recommender system for scientific literature was under development. Mendeley[1] is a service similar to Last.fmfor managing and sharing research papers, discovering research data and recommend contents to researcher. It automatically learns the user's preference by tracking his read habits [HR08]. In contrast to our solution, it utilizes a traditional client-server approach which implies high dependence on predefined central server.

6.3 Future work

With this work we showed the proof-of-concepts of a fully-decentralized search and recommendation system semantically-based for scientific literature.

Future work should pursue the state-of-the-art while extracting structured information from unstructered text document such as PDF files by employing machine learning approaches.

A machine learning approach should be employed also for learning user's information needs. This may take into account some objective measurements such as time spent on reading a paper, frequency of readings, tracking of search queries and so on.

User's profile and ranking function which are employed by the gossip-based social network management protocol should be tuned in order to achieve well-desirable properties regarding network's dynamics and topology features such as node's in-degree and out-degree.

6.3.1 Vision

P2P architectures have the potential to accelerate communication process and reduce collaboration costs through ad-hoc administration of working group. This potential could be leveraged in the world of research leading to a platform which relies on a social network of researcher which could help in collaboration. A researcher could discover many researcher with similar interests and share ideas, papers and experiences. Interesting data could be recommended by relying on reputation metrics and collaborative filtering.

[1] https://www.mendeley.com

Appendix A

Bibliography analysis

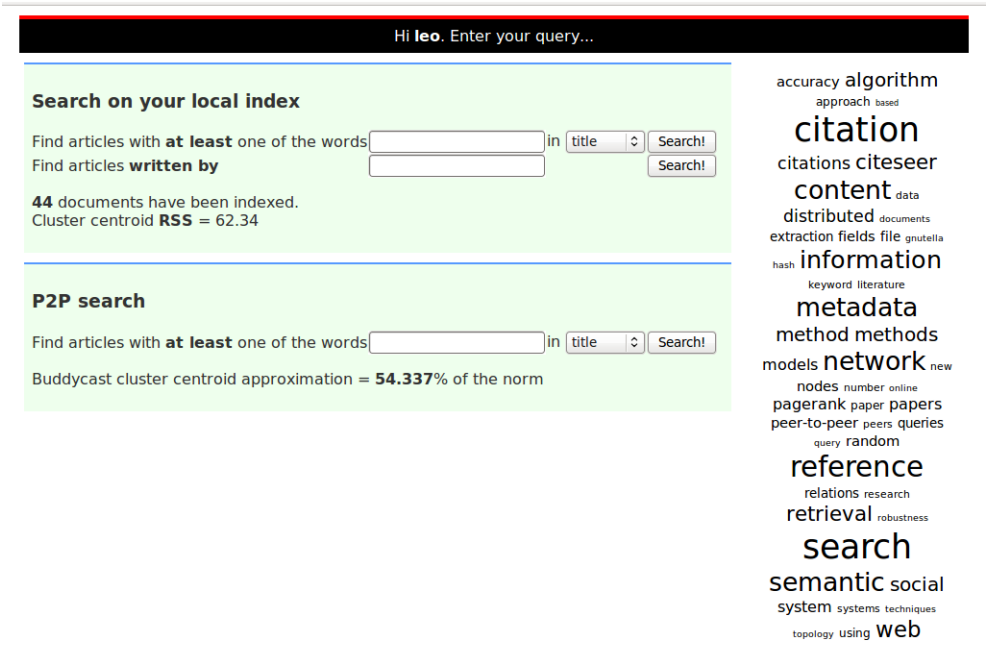


FIGURE A.1: The cloud is made up of the top-50 weighted terms, according to the tf-idf term weighting scheme (cf. section 3.1.2), of the centroid (cf. equation 3.10) of the term-by-document matrix (cf. equation 3.1) built by relying upon the vector space model of the words within the abstract section of the papers from bibliography.

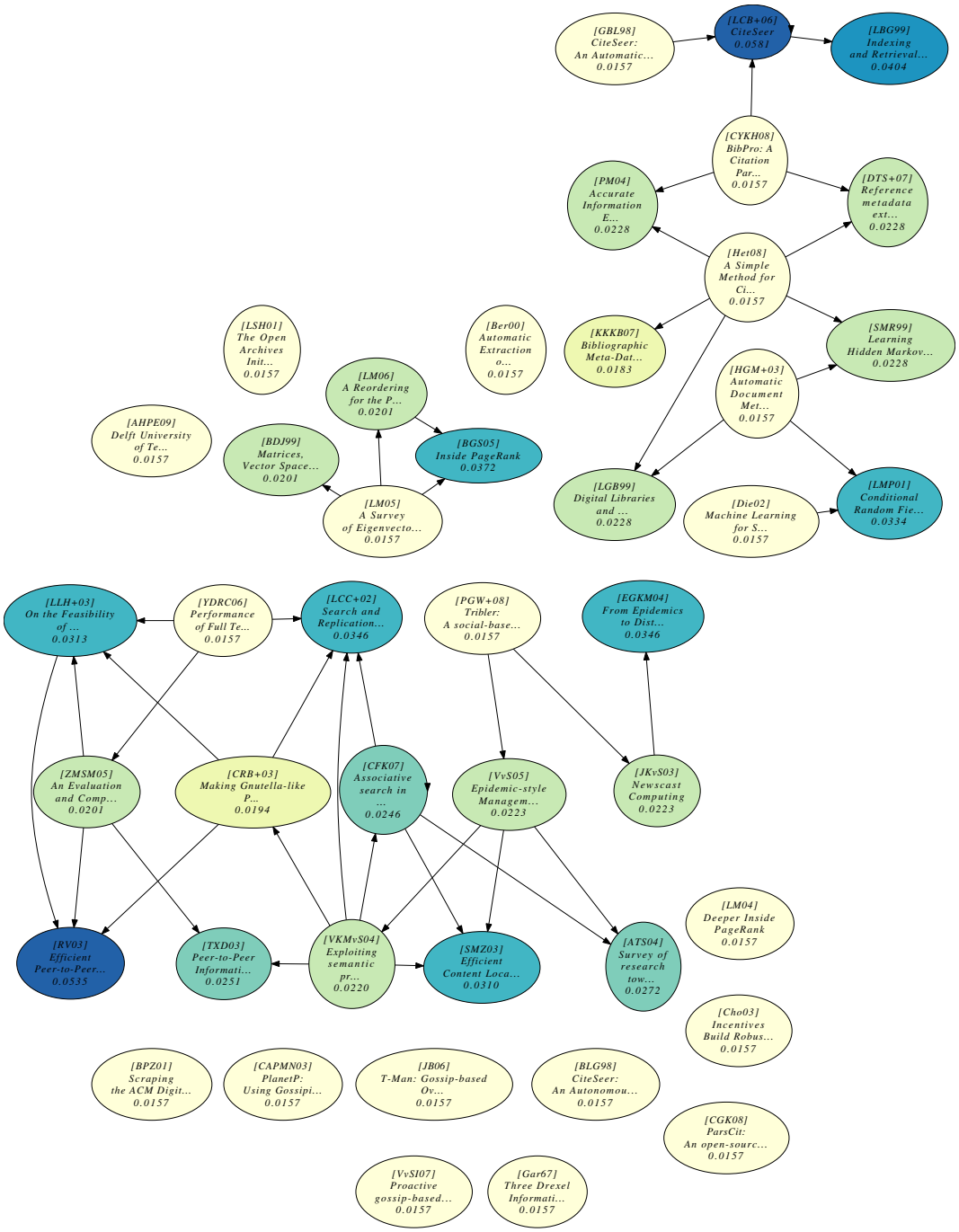


Figure A.2: PageRank analysis, performed with a perturbation coefficient $\alpha = 0.85$, of the citation graph induced by citations of papers from the bibliography (cf. section 3.4). It highlights the main topics which have been addressed in this work, i.e. 1) automatic information extraction from scientific literature; 2) mining models while analyzing scientific literature; 3) P2P distributed content location.

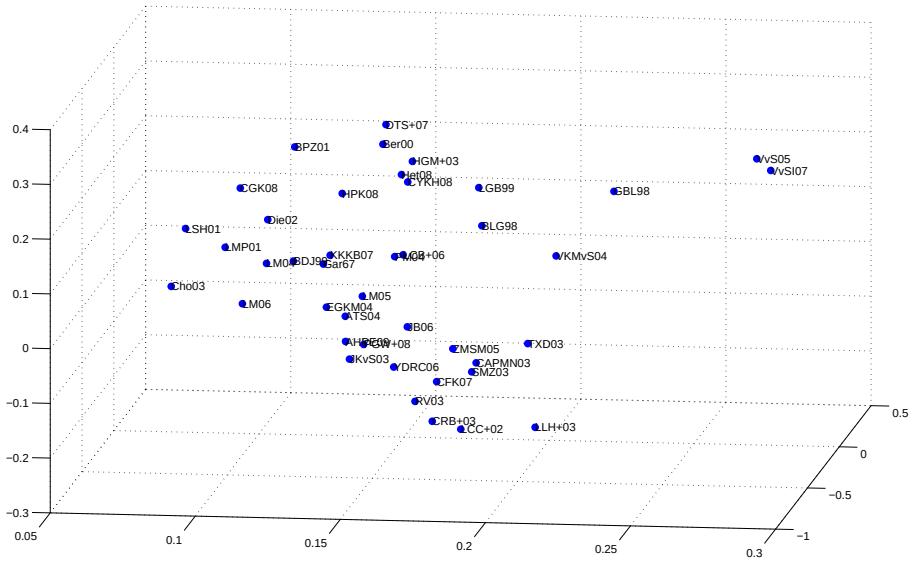


FIGURE A.3: Projection of bibliography document vectors into a 3-d semantic subspace obtained by retaining the 3 largest singular values while performing LSI (cf. section 3.2)

Bibliography

[AHPE09] Ali Abbas, David Hales, Johan Pouwelse, and Dick Epema. A gossip-based distributed social networking system. Technical Report PDS-2009-001, Delft University of Technology, 2009. I.2, 1.2

[ATS04] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371, 2004. I.1, 1.1, 4.1.2

[BDJ99] M.W. Berry, Z. Drmac, and E.R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999. 3.1, 3.2

[Ber00] D. Bergmark. Automatic extraction of reference linking information from online documents. Technical Report 2000-1821, Cornell University, Ithaca, NY, 2000. 2.2.3, 2.4.3, 2.4.3

[BGS05] M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Transactions on Internet Technology (TOIT)*, 5(1):92–128, 2005. 3.4.2

[BLG98] K.D. Bollacker, S. Lawrence, and C.L. Giles. CiteSeer: an autonous Web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the second international conference on Autonomous agents*, pages 116–123. ACM New York, USA, 1998. 2.3.2, 2.4.3

[BLH01] T. Berners-Lee and J. Hendler. Publishing on the semantic web. *Nature*, 410:1023–1023, April 2001. 2.1.1

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998. 4.1.1

[BPZ01] Donna Bergmark, Paradee Phempoonpanich, and Shumin Zhao. Scraping the acm digital library. *SIGIR Forum*, 35(2):1–7, 2001. 2.4.3

[CAPMN03] F.M. Cuenca-Acuna, C. Peery, R.P. Martin, and T.D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 2003. 4.2.5

[CdSG+07] Eli Cortez, Altigran S. da Silva, Marcos André Gonçalves, Filipe Mesquita, and Edleno S. de Moura. Flux-cim: flexible unsupervised extraction of citation metadata. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 215–224, New York, USA, 2007. 2.4.2

[CFK07] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. *Computer Networks*, 51(8):1861–1881, 2007. 4.2.5

[CGK08] I.G. Councill, C.L. Giles, and M.Y. Kan. ParsCit: An open-source CRF reference string parsing package. In *International Conference on Language Resources and Evaluation (LREC)*. Marrakech, Morocco, 2008. 2.4, 2.4.2, 2.4.2, 2.10(a), 2.10

[Cha03] S. Chakrabarti. *Mining the Web: Discovering knowledge from hypertext data*. Morgan Kaufmann, 2003. II.2, 3.2, 3.3.2, 3.4

[CLMR01] J.B. Claivaz, J.Y. Le Meur, and N. Robinson. From Fulltext Documents to Structured Citations: CERN's Automated Solution. *HEP Libraries Webzine*, 5(2), 2001. 2.4.3

[Coh03] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, volume 6, 2003. 4.1.1

[CRB+03] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA, 2003. ACM. I.2, 1.2, 4.2.5

[CYKH08] C.C. Chen, K.H. Yang, H.Y. Kao, and J.M. Ho. BibPro: A Citation Parser Based on Sequence Alignment Techniques. In *22nd International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, pages 1175–1180, 2008. 2.4.2

[DDF+90] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. 3.2.2

[Die02] T.G. Dietterich. Machine Learning for Sequential Data: A Review. *Lecture Notes in Computer Science*, pages 15–30, 2002. 2.3.2

[DTS+07] Min-Yuh Day, Richard Tzong-Han Tsai, Cheng-Lung Sung, Chiu-Chen Hsieh, Cheng-Wei Lee, Shih-Hung Wu, Kun-Pin Wu, Chorng-Shyong Ong, and Wen-Lian Hsu. Reference metadata extraction using a hierarchical knowledge representation framework. *Decision Support Systems*, 43(1):152–167, 2007. 2.4.2, 2.1, 2.4.2

[EGKM04] P.T. Eugster, R. Guerraoui, A.M. Kermarrec, and L. Massoulié. From epidemics to distributed computing. *IEEE computer*, 37(5):60–67, 2004. 4.4.1

[Gar67] R. Garner. *A computer oriented, graph theoretic analysis of Citation Index structures*. PhD thesis, Drexel University Press, Philadelphia, 1967. 2.1.2, 2.4.3

[Gar79] E. Garfield. *Citation Indexing - Its Theory and Application in Science, Technology and Humanities*. John Wiley and Sons, 1979. 2.1.2

[GBL98] C.L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An Automatic Citation Indexing System. In *Digital Libraries*, volume 98, pages 89–98, 1998. 2.3.2, 2.4.1, 2.4.2, 2.4.3

[Het08] Erik Hetzner. A simple method for citation metadata extraction using hidden markov models. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 280–284, New York, NY, USA, 2008. ACM. 2.4.2

[HGM+03] H. Han, C.L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E.A. Fox. Automatic document metadata extraction using support vector machines. In *Joint Conference on Digital Libraries*, pages 37–48, 2003. 2.3.2

[HPK08] Duncan Hull, Steve R. Pettifer, and Douglas B. Kell. Defrosting the digital library: Bibliographic tools for the next generation web. *PLoS Computational Biology*, 4(10), October 2008. I.3, 1.3

[HR08] Victor Henning and Jan Reichelt. Mendeley - A Last.fm For Research? In *Proceedings of the 4th IEEE International Conference on e-Science*, pages 327–328, Washington, DC, USA, December 2008. IEEE Computer Society. III.2, 6.2

[JB06] M. Jelasity and O. Babaoglu. T-Man: Gossip-based overlay topology management. *Lecture Notes in Computer Science*, 3910:1, 2006. 4.4.4

[JKvS03] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, November 2003. 4.4.3

[KKKB07] M. Kramer, H. Kaprykowsky, D. Keysers, and T. Breuel. Bibliographic Meta-Data Extraction Using Probabilistic Finite State Transducers. *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 2:609–613, September 2007. 2.4.2

[KM00] Taku Kudoh and Yuji Matsumoto. Use of Support Vector Learning for Chunk Identification. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 142–144. Lisbon, Portugal, 2000. 2.3.2

[LBG99] S. Lawrence, K. Bollacker, and C.L. Giles. Indexing and retrieval of scientific literature. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 139–146. ACM Press New York, NY, USA, 1999. 2.4.3

[LCB+06] Huajing Li, Isaac G. Councill, Levent Bolelli, Ding Zhou, Yang Song, Wang-Chien Lee, Anand Sivasubramaniam, and C. Lee Giles. Citeseer$^{\chi}$: a scalable autonomous scientific digital library. In *Infoscale*, page 18, 2006. 2.4.2

[LCC+02] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the* 16^{th} *international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM. 4.2.5

[LGB99] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *Computer*, 32(6):67–71, 1999. 2.3.2, 2.4.2, 2.4.3

[LLH+03] J. Li, B.T. Loo, J.M. Hellerstein, M.F. Kaashoek, D.R. Karger, and R. Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. *Lecture Notes in Computer Science*, pages 207–215, 2003. I.2, 1.2, 4.2.4

[LM04] A.N. Langville and C.D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004. 3.4.2

[LM05] A.N. Langville and C.D. Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Review*, 47(1):135–161, 2005. 3.4.2

[LM06] A.N. Langville and C.D. Meyer. A reordering for the PageRank problem. *SIAM Journal on Scientific Computing*, 27(6):2112–2120, 2006. 3.4.2

[LMP01] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *10^{th} conference on Machine Learning International Workshop*, pages 282–289, 2001. 2.4.2

[LVdS01] Carl Lagoze and Herbert Van de Sompel. The open archives initiative: building a low-barrier interoperability framework. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 54–62, New York, NY, USA, 2001. ACM. 2.2

[MH08] A. Marcozzi and D. Hales. Emergent social rationality in a peer-to-peer system. *Advances in Complex Systems (ACS)*, 11(4):581–595, 2008. I.1.1, 1.1.1

[MRS08] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. II.2, 2.5.1, 3.3.2, 3.4

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. 3.4.2, 3.4.2

[PGW+08] JA Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, DHJ Epema, M. Reinders, MR Van Steen, and HJ Sips. Tribler: a social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 20(2):127–138, 2008. I.2, III.2, 1.2, 4.4.4, 6.2

[PM04] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 329–336, 2004. 2.3.2, 2.4.2

[Por97] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997. 3.1.1, 5.4.1

[PYM+08] J.A. Pouwelse, J. Yang, M. Meulpolder, D.H.J. Epema, and H.J. Sips. Buddycast: an operational peer-to-peer epidemic protocol stack. In *Proc. of the 14^{th} Annual Conf. of the Advanced School for Computing and Imaging*, pages 200–205. ASCI, 2008. 4.4.4

[RM06] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 50(17):3485–3521, 2006. I.2, 1.2, 4.1.2

[RV03] P. Reynolds and A. Vahdat. Efficient Peer-to-Peer Keyword Searching. *Lecture Notes in Computer Science*, pages 21–40, 2003. I.2, 1.2, 4.2.4

[SMR99] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999. 2.3.2

[SMZ03] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In 22^{nd} *Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2166–2176, April 2003. 4.3.1, 4.3.1

[TXD03] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 175–186. ACM New York, NY, USA, 2003. 4.3

[VGVS05] S. Voulgaris, D. Gavidia, and M. Van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005. 4.4.3

[VKMvS04] Spyros Voulgaris, Anne-Marie Kermarrec, Laurent Massoulié, and Maarten van Steen. Exploiting semantic proximity in peer-to-peer content searching. In 10^{th} *IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 238–243, 2004. 4.3.1

[VvS05] S. Voulgaris and M. van Steen. Epidemic-Style Management of Semantic Overlays for Content-Based Searching. *Lecture Notes in Computer Science*, 3648:1143, 2005. I.2, 1.2, 4.4.4, 4.4.4

[VvSI07] S. Voulgaris, M. van Steen, and K. Iwanicki. Proactive gossip-based management of semantic overlay networks. *Concurrency and Computation*, 19(17):2299, 2007. 4.4.4, 4.4.4

[YDRC06] Y. Yang, R. Dunlap, M. Rexroad, and B.F. Cooper. Performance of full text search in structured and unstructured peer-to-peer systems. In *IEEE INFOCOM*, pages 2658–2669. IEEE Press, 2006. 4.2.4

[ZMSM05] M. Zhong, J. Moore, K. Shen, and A.L. Murphy. An evaluation and comparison of current peer-to-peer full-text keyword search techniques. In *Proceedings of the* 8^{th} *International Workshop on the Web & Databases (WebDB2005), ACM SIGMOD/PODS 2005 Conference*, 2005. 4.1.2, 4.2.4