UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

STUDIO E SVILUPPO DI ALGORITMI DI CONTEXT REASONING PER DISPOSITIVI MOBILI ED INTEGRAZIONE NEL MIDDLEWARE CAMEO

Relatore: Prof. Elena PAGANI
Correlatori: Dott. Franca DELMASTRO
Dott. Valerio ARNABOLDI

Tesi di Laurea di:
Mattia Giovanni CAMPANA
Matr. Nr. 805165

ANNO ACCADEMICO 2013-2014

A mia madre, la persona a cui devo tutto.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

I continui progressi tecnologici stanno portando, ogni giorno di più, verso un mondo ricco di sensori (mobili e statici), smartphone e dispositivi indossabili con capacità computazionali sempre più avanzate. L'utilizzo massiccio di questi dispositivi nella vita quotidiana ha permesso di rafforzare sempre di più il legame esistente tra il *mondo fisico* e quello *virtuale* (applicazioni, dispositivi e servizi). Tale fenomeno viene indicato in letteratura come *convergenza del mondo fisico e virtuale (CPW, Cyber-Phisical World)* [20]. Sfruttando i dispositivi citati e varie tecnologie, le informazioni riguardanti la realtà fisica (raccolte ad esempio tramite l'utilizzo di sensori) vengono direttamente trasferite nel mondo virtuale, all'interno del quale possono essere elaborate al fine di adattare le applicazioni e i servizi al contesto fisico corrente, e, tramite l'ausilio di attuatori, modificare o adattare possibilmente lo stesso ambiente fisico. In altre parole, le azioni e le informazioni prodotte nel mondo fisico sono in grado di modificare i contesti sociale e personale degli utenti, i quali, di conseguenza, possono avere effetto sul modo in cui i servizi elaborano le informazioni nel mondo virtuale.

Chiaramente, nello scenario fin qui delineato, gli esseri umani ricoprono un ruolo centrale: ogni persona ha accesso a svariati dispositivi dotati di differenti capacità computazionali e connettività, attraverso i quali ha la possibilità di interagire con il mondo virtuale, creando così un collegamento tra il proprio ambiente fisico e il mondo elettronico. Le relazioni sociali delle persone, inoltre, svolgo-

no un ruolo di primaria importanza nel modo in cui le informazioni circolano nel mondo virtuale e condizionano la realizzazione di applicazioni e servizi pervasivi. In Figura 1.1 viene mostrato come, grazie al fenomeno di convergenza del mondo fisico-virtuale, le relazioni sociali umane, fisiche (tra i dispositivi) e virtuali [1] siano sempre più interconnesse tra loro.

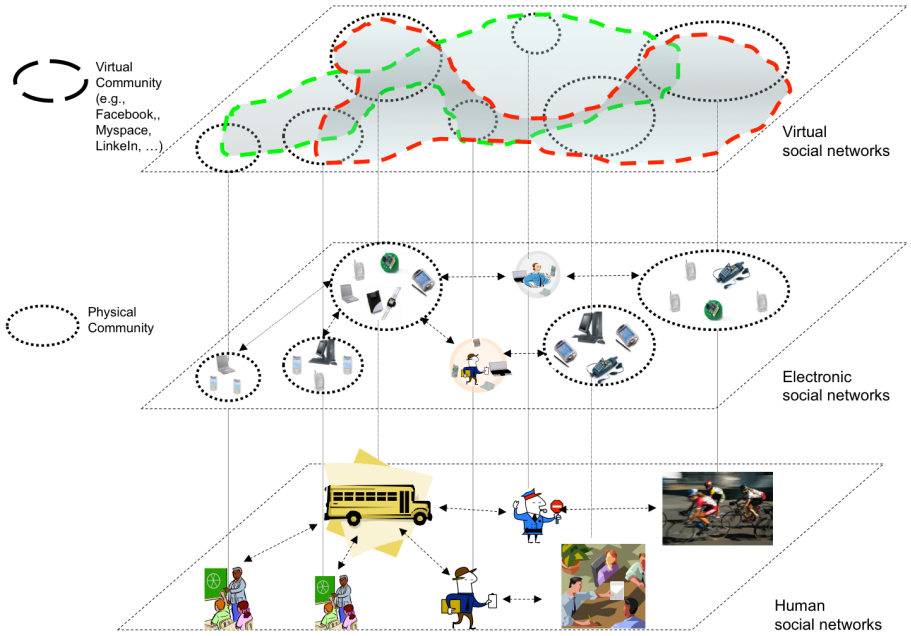


Figura 1.1: Interconnessione delle reti sociali umane, fisiche e virtuali [6].

In questo scenario, i cosiddetti dispositivi *smart* (smartphone, sensori, wearable, etc...) possono costituire una densa infrastruttura per monitorare l'ambiente circostante e collezionare informazioni relative ai comportamenti delle persone, le loro necessità e le dinamiche che governano le relazioni sociali. Queste informazioni possono circolare tra i dispositivi sfruttando la loro connettività wireless e le interazioni fisiche che avvengono tra gli utenti; tale concetto rappresenta la chiave delle reti opportunistiche. In ambiente opportunistico la rete si forma direttamente tra i dispositivi degli utenti in condizioni di connettività intermittente. Il principio di base è quello di sfruttare tutte le opportunità di comunicazione

[1] Per relazioni sociali virtuali si intendono i rapporti tra le persone esistenti negli Online Social Network come Facebook, Linkedin, Twitter, etc...

offerte dalle tecnologie wireless disponibili sui singoli dispositivi e di utilizzare "opportunisticamente" la mobilità degli utenti per instaurare delle comunicazioni attraverso cui condividere informazioni, contenuti e risorse. Le reti opportunistiche rappresentano una delle più interessanti evoluzioni delle MANET (Mobile Ad-hoc NETwork). A differenza di quanto accade per le reti ad-hoc, nelle reti opportunistiche si assume che non esista mai un path che colleghi due nodi intenzionati a comunicare. Seppur entrambi possano non essere mai connessi alla stessa rete nel medesimo istante, le tecniche per reti opportunistiche permettono ad essi di comunicare tra loro scambiandosi messaggi.

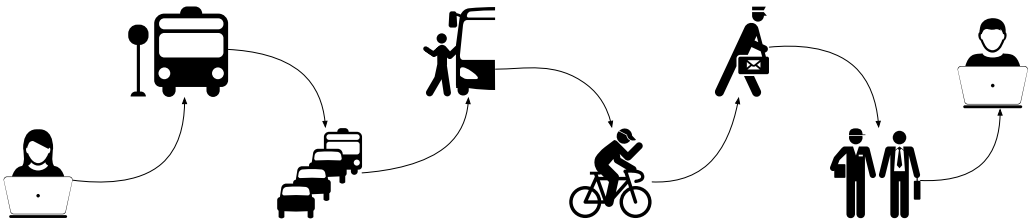


Figura 1.2: Esempio di rete opportunistica [51].

In Figura 1.2 viene riportato un esempio esplicativo del concetto alla base delle reti opportunistiche: una donna invia opportunisticamente a un suo amico un messagio dal proprio computer desktop tramite un collegamento Wi-Fi. Il messaggio viene ricevuto da un autobus che sta transitando nei pressi della donna, il quale ha la possibilità di trasportarlo verso la destinazione. Tramite link Bluetooth, l'autobus inoltra il messaggio allo smartphone di una ragazza che sta scendendo ad una fermata vicino a un parco per andare in università. Lo smartphone della ragazza entra in contatto con il dispositivo di un ciclista e, dal momento che si sta dirigendo verso la destinazione, gli invia il messaggio. Continuando nello stesso modo, dopo qualche hop, il messaggio, eventualmente, giungerà finalmente a destinazione. Come è chiaro da questo semplice esempio, un collegamento diretto tra le due persone non esiste mai, ma, sfruttando opportunisticamente i contatti tra dispositivi eterogenei, il messaggio viene distribuito hop dopo hop (sperabilmente) vicino alla destinazione ed eventualmente consegnato a quest'ultima.

In [14] viene dimostrato come, nelle reti opportunistiche, sia possibile ottimizzare la distribuzione dei contenuti basandosi su informazioni di contesto come interessi, abitudini e relazioni sociali degli utenti.

Queste informazioni possono derivare dalle letture effettuate dai vari sensori oppure essere dedotte da altri dati di contesto. Si rendono quindi necessari dei meccanismi in grado di inferire nuove e rilevanti informazioni ad uso delle applicazioni e degli utenti. In letteratura, tali meccanismi, prendono il nome di *context-reasoning*.

A questo scopo l'Istituto di Informatica e Telematica (IIT) del Consiglio Nazionale delle Ricerche (CNR) [2] ha sviluppato CAMEO [6], un middleware in fase prototipale per sistemi mobili e pervasivi, che consente ad ogni dispositivo personale dell'utente di raccogliere ed elaborare informazioni di contesto che identificano lo stato del dispositivo, il profilo dell'utente e il contesto sociale in cui si trova.

In questo scenario è importante determinare l'utilità ed il livello di interesse per l'utente locale di un contenuto o una risorsa disponibile sulla rete opportunistica.

L'obiettivo di questo lavoro di tesi è quello di definire un algoritmo di reasoning che, sfruttando informazioni di contesto relative a contenuti e/o risorse disponibili, e gli interessi di uno o più utenti, sia in grado di calcolare l'utilità dei contenuti (o risorse) per il nodo locale, in base ad una metrica di "interesse".

In una rete opportunistica un nodo non ha mai la visione completa delle informazioni che circolano in tutta la rete, ma soltanto quella relativa alle proprie informazioni e quelle dei suoi vicini. L'algoritmo dovrà quindi essere in grado di prendere decisioni analizzando un set di informazioni parziali che rispecchiano la natura frammentaria della rete opportunistica.

Prendendo come riferimento il problema della distribuzione dei contenuti in ambiente opportunistico, è possibile assumere che ad ogni contenuto siano associati dei *tag* che lo caratterizzano, definiti dalle applicazioni o dagli utenti stessi.

[2] http://www.iit.cnr.it

La pratica di permettere agli utenti di assegnare liberamente dei *tag* alle risorse dà luogo a quella che in letteratura viene chiamata *folksonomia* [43], ovvero una tassonomia creata da chi la utilizza in base a criteri individuali.

In una folksonomia non esiste a priori un'ontologia esplicita che collega i contenuti ed i tag associati. L'algoritmo proposto dovrà quindi essere in grado di scoprire i legami semantici tra i contenuti, analizzando i loro tag, e la relazione con gli interessi dell'utente in termini di "similarità tra contenuti" e "utilità per gli utenti".

In ambiente opportunistico la folksonomia sarà caratterizzata da un'elevata dinamicità dovuta alla mobilità dei nodi e dai conseguenti partizionamenti della rete.

L'analisi delle folksonomie è ampiamente diffusa nei sistemi Web per la definizione di *Tag-Based Recommender System* [63]. In modo analogo a questi sistemi, la soluzione proposta si basa sull'utilizzo di grafi che legano tra loro utenti, contenuti e tag. Un grafo di questo tipo viene qui definito come *"grafo di conoscenza"*. A differenza dei sistemi Web, in cui si ha una conoscenza completa del grafo sulla rete, in una rete opportunistica ogni nodo ha una visione parziale del grafo di conoscenza costruita in base alle informazioni locali e rilevate dai nodi incontrati. L'utilizzo di un approccio basato sui grafi, noto in letteratura come *"network-based"* [63], risulta particolarmente efficiente per lo scenario prefissato e, inoltre, lascia spazio per future analisi di tipo graph-based, come, ad esempio, algoritmi di community detection context-based.

In una rete opportunistica questo approccio può essere usato per vari scopi, come ad esempio:

1. identificare contenuti disponibili in rete e di possibile interesse per l'utente locale
2. evidenziare affinità di interessi tra utenti in prossimità
3. ottimizzare la condivisione di risorse

La novità del lavoro consiste nella definizione di un nuovo algoritmo per tag-based recommeder system che migliora le prestazioni rispetto allo stato dell'arte

in termini di precisione nella selezione dei contenuti da suggerire in base agli interessi dell'utente. Inoltre, lo stesso, viene applicato per la prima volta nelle reti opportunistiche come algoritmo di reasoning per la realizzazione di servizi di condivisione di contenuti e/o risorse context- e social-aware.

L'algoritmo proposto in questa tesi, *PopuLarity-based ItEm Recommender System (PLIERS)*, seleziona i contenuti da suggerire privilegiando quelli con popolarità simile a quella degli item già in possesso dell'utente locale. Questo comportamento lo differenzia dagli algoritmi presenti in letteratura [63] che prediligono contenuti con massima o minima popolarità rispetto agli interessi dell'utente.

In ambiente opportunistico, il grafo che rappresenta la relazione tra un utente ed i contenuti che possiede o genera nel tempo può rappresentare un'ulteriore informazione di contesto locale che viene scambiata tra i nodi quando si incontrano, sfruttando la politica di raccolta delle informazioni sviluppata in CAMEO. Ogni nodo, quindi, mantiene solo una visione parziale del grafo di conoscenza e, ad ogni contatto, scambia questo grafo con i propri vicini, cercando di ricostruire una visione globale delle informazioni in rete. Il grafo rappresenta la relazione tra un utente ed i contenuti che possiede o genera nel tempo. La dinamicità del grafo è quindi legata sia alla dinamicità degli utenti nella generazione dei contenuti, sia ai continui aggiornamenti dei dati di contesto effettuati da CAMEO. Ogni volta che, tramite CAMEO, viene scoperto un nuovo contenuto disponibile in prossimità, la sua utilità verrà valutata tramite PLIERS per effettuarne il download automatico o suggerirlo all'utente.

Dato che nella rete opportunistica i nodi hanno risorse limitate e i contatti sono in genere di breve durata, è necessario che l'algoritmo di reasoning risulti essere efficiente sia dal punto di vista dell'occupazione di memoria, sia in termini di tempo necessario alla sua esecuzione.

Per valutare l'efficacia dell'algoritmo in ambiente opportunistico sono state eseguite diverse simulazioni, le quali hanno permesso di verficare quanto il grafo di conoscenza costruito da ciascun nodo differisca da quello globale, costruito a priori sulla base di dati derivati dalla rete sociale Twitter. In aggiunta, le simulazioni hanno permesso di scoprire quanti contatti opportunistici siano necessari

prima che un nodo riesca ad ottenere una buona approssimazione dell'intero grafo di conoscenza.

Inoltre, simulando la generazione di contenuti nella rete, è stato possibile misurare la precisione con cui i contenuti vengano selezionati rispetto alla visione globale del grafo di conoscenza.

Il lavoro di tesi è così organizzato: nel Capitolo 2 vengono descritte le componenti principali dell'architettura del middlweare CAMEO, riportando in particolare le informazioni considerate per la rappresentazione del contesto e come vengono modellate. Il Capitolo 3 definisce il concetto di context-reasoning, descrivendo le principali tecniche presenti in letteratura per inferire sui dati di contesto. Vengono inoltre riportate le tecniche attualmente utilizzate in CAMEO e presentata una soluzione modulare allo scopo di estendere le capacità di reasoning del middleware. Nel Capitolo 4 viene affrontato il problema della distribuzione dei contenuti basata su informazioni di contesto, introducendo le tecniche presenti in letteratura per la realizzazione di Tag-based Recommender System e analizzando in dettaglio le soluzioni ritenute più efficienti per lo scenario di riferimento. Nel Capitolo 5 viene descritta la soluzione proposta per tale problema: PLIERS. Viene inoltre descritta la sua applicazione in ambiente opportunistico, la tecnica utilizzata per renderlo adattivo ai contenuti che vengono condivisi in rete ed i metodi proposti per evitare una crescita incontrollata del grafo di conoscenza locale. Il Capitolo 6 contiene i risultati ottenuti dai test svolti per la validazione dell'algoritmo e le simulazioni effettuate per dimostrarne l'efficacia in ambiente opportunistico. Infine, nel Capitolo 7, vengono riportate le conclusioni e gli sviluppi futuri del lavoro svolto.

Capitolo 2

CAMEO

Negli ultimi anni la capacità dei sistemi mobili e pervasivi di reagire e adattarsi ai cambiamenti di contesto è diventata sempre più un requisito fondamentale per migliorare l'impatto che hanno le nuove soluzioni tecnologiche sulla qualità di vita dei singoli individui, dei gruppi di persone che condividono interessi e abitudini e, da un punto di vista più ampio, dell'intera società.

Di recente sono emersi nuovi domini applicativi per i sistemi mobili e pervasivi orientati a migliorare le condizioni di benessere delle persone e della società (sistemi di personal healthcare, monitoraggio ambientale, etc...). Questi domini sono caratterizzati da un forte dinamismo dovuto principalmente alla mobilità degli utenti, interoperabilità dei dispositivi, interazioni delle applicazioni con l'ambiente esterno ma anche dalle interazioni tra gli utenti e i loro stessi dispositivi. Tali condizioni contribuiscono significamente al fenomeno di "convergenza del mondo virtuale-fisico" (*cyber-physical world - CPW -*) descritto nel Capitolo 1 in cui gli essere umani sono posti al centro del processo, sia come attori che come spettatori. Infatti, le loro azioni e informazioni generate nel "mondo fisico" posso avere effetto (e modificare) sui loro contesti sociale e personale, influenzando di conseguenza il modo in cui le informazioni e i servizi vengono gestiti nel "mondo virtuale". Nel contempo, le interazioni sociali e lo scambio di informazioni nel "mondo virtuale" possono influenzare il comportamento degli utenti nel "mondo fisico".

In questo scenario la nozione di *contesto* per i sistemi mobili e pervasivi deve essere ampliata in modo tale da includere le condizioni sia dell'ambiente esterno sia quelle sociali, così da creare dei sistemi automatici e auto-adattivi basati sulle informazioni personali dell'utente.

In linea con gli obiettivi di ricerca presentati in [20], l'Istituto di Informatica e Telematica (IIT) del Consiglio Nazionale delle Ricerche (CNR) ha sviluppato CAMEO [6], un middleware in fase prototipale per sistemi mobili e pervasivi, caratterizzato dalla capacità di raccogliere e condividere informazioni di contesto multidimensionale derivate sia dal "mondo fisico" che dal "mondo virtuale". CAMEO ha lo scopo di supportare lo sviluppo di un nuovo tipo di applicazioni mobili chiamate *mobile social network (MSN)*, orientate a migliorare l'esperienza d'uso degli utenti, le loro interazioni sociali e di stimolare il concetto di *collective awareness* (interesse verso la collettività) utilizzando il proprio dispositivo mobile.

Le applicazioni MSN ereditano le stesse funzionalità di condivisione e scambio di informazioni tipiche degli *online social network (OSN)*, estendendo però i loro scenari applicativi sfruttando le opportunità di interazione fisica tra i dispositivi generate dalla mobilità degli utenti e la comunicazione opportunistica. In un certo senso, i mobile social network superano le capacità degli OSN, generalmente basati su relazioni sociali preesistenti (amici o amici di amici), stimolando nuove interazioni sociali tra utenti che non si conoscono a priori ma che condividono interessi comuni. Gli MSN sono allo stesso momento incentrati sulle persone, mettendo in relazione le interazioni sociali degli essere umani con la rete di dispositivi elettronici, e incentrati sui contenuti, mettendo a disposizione un metodo efficiente per la disseminazione di contenuti in rete in linea con gli interessi degli utenti.

CAMEO permette ai dispositivi mobili, che occasionalmente si trovano fisicamente a distanza ravvicinata, di scoprire automaticamente gli interessi in comune tra gli utenti, i servizi e, in un'ottica di *sensing partecipativo* [1], anche le risorse messe a disposizione tramite la comunicazione opportunistica.

In questo capitolo vengono descritte le principali caratteristiche di CAMEO

facendo riferimento principalmente a [5] e [6]. Innanzitutto viene analizzato come è rappresentato il contesto in CAMEO e da quali informazioni è composto; in secondo luogo viene riportata la struttura architetturale del middleware e le sue componenti principali e, infine, l'inerfaccia di comunicazione tra CAMEO e le applicazioni poste al livello più alto dell'architettura (le applicazione MSN).

2.1 Well-being context

CAMEO introduce la nozione di *well-being context* unificando così due caratteristiche essenziali dei moderni sistemi mobili e pervasivi: quella di essere allo stesso momento sensibile sia ai cambiamenti di contesto (*context-awareness*) sia a quelli inerenti le relazioni sociali dell'utente (*social-awareness*). Correlando le informazioni locali riguardanti l'utente e il suo dispositivo (es. profilo utente, interessi, attività, risorse locali, etc ...) con le informazioni derivate da altri utenti e dispositivi, CAMEO è in grado di definire nuovi pattern di comunicazione fisici e nuove interazioni sociali tra gli utenti. In questo modo le informazioni di contesto possono essere usate per ottimizzare sia i servizi di basso livello (networking e condivisione di risorse), sia le applicazioni MSN di alto livello.

Il contesto di un *nodo* (coppia utente-dispositivo) è definito dall'integrazione di tre componenti principali (Figura 2.1): il contesto locale (*local context*), il contesto esterno (*external context*) e il contesto sociale (*social context*).

Il *local context* rappresenta tutte le informazioni riguardanti l'utente e il suo dispositivo mobile; nella fattispecie contiene:

User's personal profile Tutte le informazioni che possono descrivere al meglio gli interessi e le abitudini dell'utente come: tipologia e luogo di lavoro, abitudini, stile di vita, appuntamenti, etc ... Tali informazioni vengono tipicamente inserite direttamente dall'utente tramite apposite interazioni con il proprio dispositivo e applicazioni in esso installate (rubrica, calendario, client di OSN, etc...).

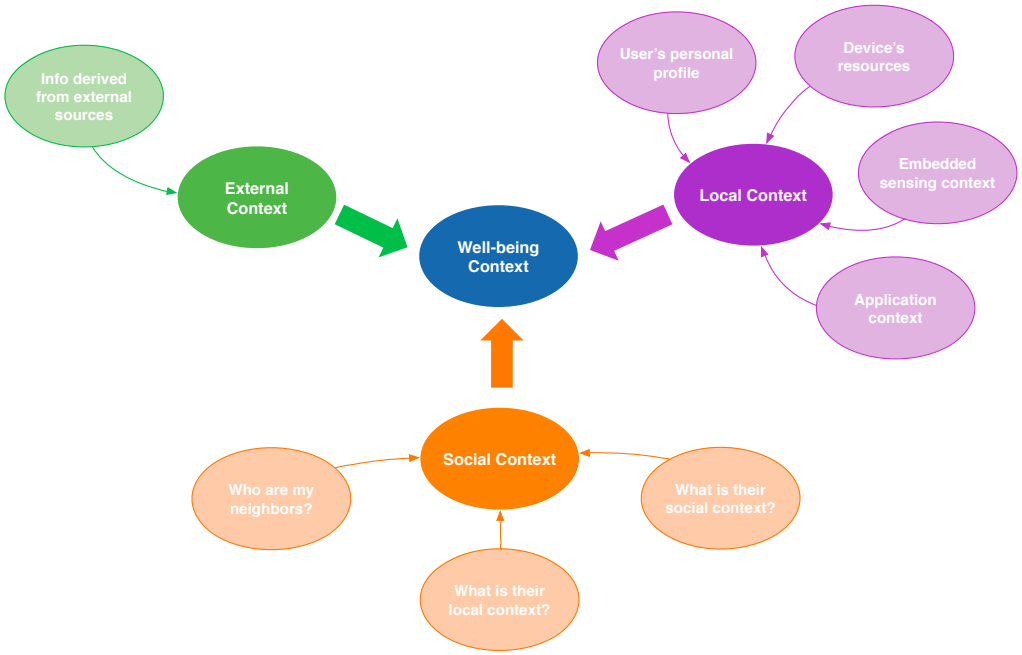


Figura 2.1: Il Well-being context di CAMEO [6].

Device's resources Informazioni riguardanti le risorse locali del dispositivo: livello della batteria, capacità della memoria, utilizzo della CPU, processi in esecuzione, connessioni di rete disponibili, sensori ad esso collegati, etc...

Embedded sensing context Informazioni raccolte dai sensori presenti nel dispositivo e utilizzate tipicamente per riconoscere le attività e posizione dell'utente. Rientrano in questo caso: GPS, accelerometri, giroscopi, bussole, fotocamere, microfoni, etc...

Application context Informazioni di contesto specificate direttamente dalle applicazioni in esecuzione sul dispositivo. Queste informazioni possono essere, ad esempio, specifiche caratteristiche associate ai contenuti gestiti da un'applicazione (tipo del contenuto o *tag* generati dagli utenti), oppure parametri utili alla corretta esecuzione dell'applicazione sul dispositivo mobile (es. risorse necessarie).

L'*external context* rappresenta l'insieme di informazioni raccolte dal disposi-

tivo tramite interazioni dirette con sorgenti esterne ad esso. Queste sorgenti di informazioni possono essere rappresentate da stazioni di sensing fisse e mobili (tra queste rientrano anche i dispositivi indossabili - *wearable* -), con lo scopo di monitorare specifici parametri (es. inquinamento ambientale, livello del traffico, battito cardiaco, etc...), oppure servizi remoti dedicati alla raccolta dati da una o più reti di sensori. Tali informazioni potranno poi essere integrate e correlate a specifici componenti del *local context*, come le informazioni raccolte dai sensori locali presenti nel dispositivo o ai contenuti generati dalle applicazioni. In questo modo il sistema è in grado di effettuare il cosiddetto *participatory and opportunistic sensing* con lo scopo di aumentare la quantità di informazioni correlate a specifici eventi e, possibilimente, di migliorarne l'accuratezza.

Nel *social context* vengono mantenute le informazioni riguardanti le interazioni fisiche del nodo locale con gli altri in prossimità; queste informazioni definiscono l'appartenenza del nodo locale a una particolare *comunità fisica*. Ogni volta che due nodi si incontrano avviene lo scambio e la raccolta di informazioni appartenenti ai reciproci *local context*. Dal momento che il *local context* di ciascun nodo contiene informazioni riguardanti sia le persone che i contenuti (abitudini dell'utente, profilo personale, interessi, tipi di contenuti, etc...), il *social context* identifica anche l'appartenenza del nodo locale a una o più *comunità virtuali*. Ad esempio, un turista che visita Roma durante le vacanze di Natale fa parte della comunità virtuale dei turisti che dichiarano di essere interessati a Roma e, nello stesso momento, alla comunità fisica degli utenti che stanno visitando Roma nello stesso periodo e che sono in prossimità dell'utente.

2.2 Context Model

La scelta di come modellare le informazioni di contesto è fondamentale per un middleware come CAMEO perchè potrebbe influenzare le performance del sistema. Le criticità possono essere rappresentate sia in termini di overhead computazionale ma, sopratutto, in termini di tempi di risposta per valutare il contesto in presenza di informazioni multidimensionali ed eterogenee come quelle contenute

nel well-being context di CAMEO. Queste rappresentano delle limitazioni sia per la gestione locale del contesto, sia per quanto riguarda la comunicazione opportunistica. Infatti, dal momento che la durata dei contatti opportunistici dipendono strettamente dalla mobilità dell'utente, dei tempi di processing delle informazioni di contesto troppo elevati possono causare la perdita di un'occasione per lo scambio di dati tra i dispositivi.

In letteratura sono state proposte diverse tecniche per la modellazione delle informazioni di contesto e in [10] è possibile trovarne una comparazione in termini di vari fattori: efficienza di accesso ai dati, supporto a procedure di *reasoning*, scalabilità e usabilità dei formalismi.

Gli approcci più semplici si basano su coppie di chiave-valore per rappresentare il contesto, ottenendo così un modello facile da implementare e con basso costo computazionale di gestione. Le critiche che sono state sollevate nei confronti di questo tipo di modelli sono legate principalmente ai loro limiti nel: rappresentare tipi e relazioni complesse tra entità, rappresentare una successione temporale degli eventi, impossibilità di esprimere un livello qualitativo delle informazioni (pensando, ad esempio, alle approssimazioni ed errori di misurazioni derivanti dai sensori), mancanza di un controllo di consistenza e la mancanza di supporto al reasoning, soprattutto in presenza di informazioni incerte (ad esempio, misurazioni di sensori contrastanti tra loro). Il rappresentante di questa classe di modelli è sicuramente il *Composite Capabilities/Preference Profile (CC/PP)*[1], lo standard W3C usato originariamente per descrivere le caratteristiche dei dispositivi mobili e che, utilizzando il *Resource Description Framework (RDF)*[2], offre la possibilità di specificare vincoli elementari e limitate relazioni tra i tipi di entità che caratterizzano il contesto.

Gli approcci ontologici possono essere visti come un'evoluzione di quelli chiave-valore con l'introduzione di descrizioni di concetti e relazioni per aggiungere un significato semantico ai dati di contesto. Le logiche descrittive a supporto delle ontologie rappresentano degli utili strumenti al reasoning ma il loro utilizzo

[1] http://www.w3.org/Mobile/CCPP
[2] http://www.w3.org/RDF

in dispositivi mobili presenta dei problemi dal punto di vista dei costi computazionali, scalabilità e tempi di processing all'aumentare delle informazioni contenute nel modello.

L'approccio scelto in CAMEO per modellare i dati di contesto rientra nella classe dei modelli *object-role based* e più precisamente nell'utilizzo del *Context Modeling Language (CML)*, inizialmente descritto in forma preliminare in [30] e rifinito successivamente in [28, 29]. CML mette a disposizione delle regole e costrutti per rappresentare *object type* (intesi come tipi di entità) e *fact type* (concettualmente simili a relazioni tra entità). Un object type viene caratterizzato in accordo alle origini del dato o al tipo di persistenza e può quindi essere: *profiled* se l'informazione è stata direttamente inserita dall'utente, *sensed* come, ad esempio, la lettura dei sensori, *derived* se derivato da altri dati di contesto, *static* oppure *dynamic*. I dati modellati possono inoltre essere arricchiti da metadati che indicano l'accuratezza e la "freschezza" (tempo trascorso dall'ultimo update) delle informazioni in essi contenuti, la cardinalità e dipendenze tra fact type e mantenere uno storico a supporto di analisi temporali dell'evoluzione del contesto. Uno degli aspetti più interessanti di CML è la possibilità di mappare lo schema concettuale con esso definito in uno schema relazione e, di conseguenza, in database relazionali. A supporto del *context-reasoning*, CML offre inoltre la possibilità di valutare semplici asserzioni espresse come semplici query SQL-Like introducendo una logica a tre valori per permettere interrogazioni basate su dati incerti. Ad esempio, la valutazione dell'asserzione *"L'utente A si trova nel luogo X"* può restituire i valori *true* o *possibly true* se uno o più luoghi sono associati al soggetto in questione. Il modello CML permette quindi un'alta espressività delle informazioni di contesto (comparabile con quello offerto dalle ontologie) supportando anche l'implementazione di efficienti tecniche di reasoning.

La Figura 2.2 mostra il modello CML del well-being context di CAMEO: gli object type sono rappresentati graficamente da ellissi e i fact type da riquadri con relative caratteristiche specificate in legenda. I differenti colori degli object type servono per evidenziare la loro appartenenza alle tre componenti principali del well-being context (local, external e social). Come è facile notare, il fulcro del

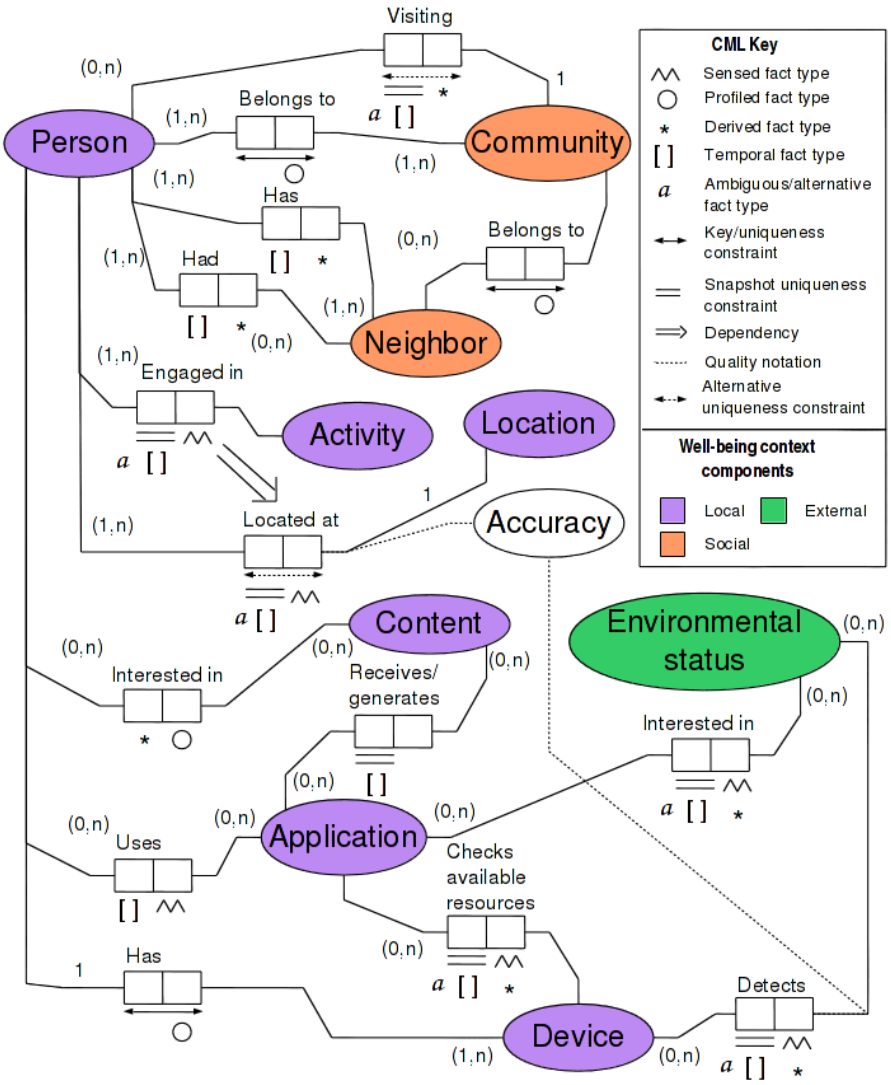


Figura 2.2: Modello CML del well-being context di CAMEO [6].

modello si riferisce al local context e, in particolar modo, si concentra principalmente sull'object type *Person*. Esso include infatti tutte le informazioni associate all'utente locale. L'entità *Person* è coinvolta in diversi *fact type* che descrivono le attività e posizioni dell'utente, le sue interazioni con il dispositivo e le applicazioni in esecuzione, e i suoi interessi in tipologie di contenuti associati a specifiche

applicazioni. Infatti il modello è stato disegnato appositamente per supportare un numero arbitrario di applicazioni eseguite sul medesimo dispositivo (e dallo stesso utente) che non necessariamente appartengono allo stesso dominio applicativo ma che possono essere interessate alle medesime informazioni di contesto; in questo modo il modello permette inoltre il riutilizzo del contesto per applicazioni diverse.

L'object type *Application* rappresenta il punto di collegamento tra le componenti dell'external context e quelle del local context. Infatti il fact type *"Application interested in Environmental status"* rappresenta la relazione tra specifiche applicazioni e le informazioni derivanti da servizi di monitoraggio ambientale esterni (web service remoti o sensori esterni al dispositivo); mentre il fact type *"Device detects Environmental status"* rappresenta la capacità del dispositivo locale di raccogliere e processare misurazioni relative all'ambiente circostante.

Per quanto riguarda le componenti che si riferiscono al social context, l'object type *Neighbor* e i fact type ad esso collegati rappresentano i dati che derivano dallo scambio di informazioni di contesto con i nodi in prossimità di quello locale. *Community* rappresenta invece le informazioni riguardanti le comunità ("home" e quelle visitate) a cui appartengono sia il nodo locale che quelli incontrati, e sono a supporto dell'algoritmo di community detection implementato in CAMEO.

2.3 Architettura

CAMEO è caratterizzato da un'architettura modulare (Figura 2.3) formata da due componenti principali:

Local resource management framework (LRM-Fw) Si occupa di interagire direttamente con le risorse locali del dispositivo sia per quanto riguarda l'hardware (come i sensori embedded), sia dal punto di vista software (primitive di rete e altre componenti del Sistema Operativo). Gestisce inoltre le interazioni tra il nodo e le possibili risorse remote (sensori esterni, reti di sensori o repository remoti).

Context-Aware Framework (CA-Fw) Si occupa di immagazzinare e processare tutte le informazioni di contesto raccolte dalle varie componenti del middleware.

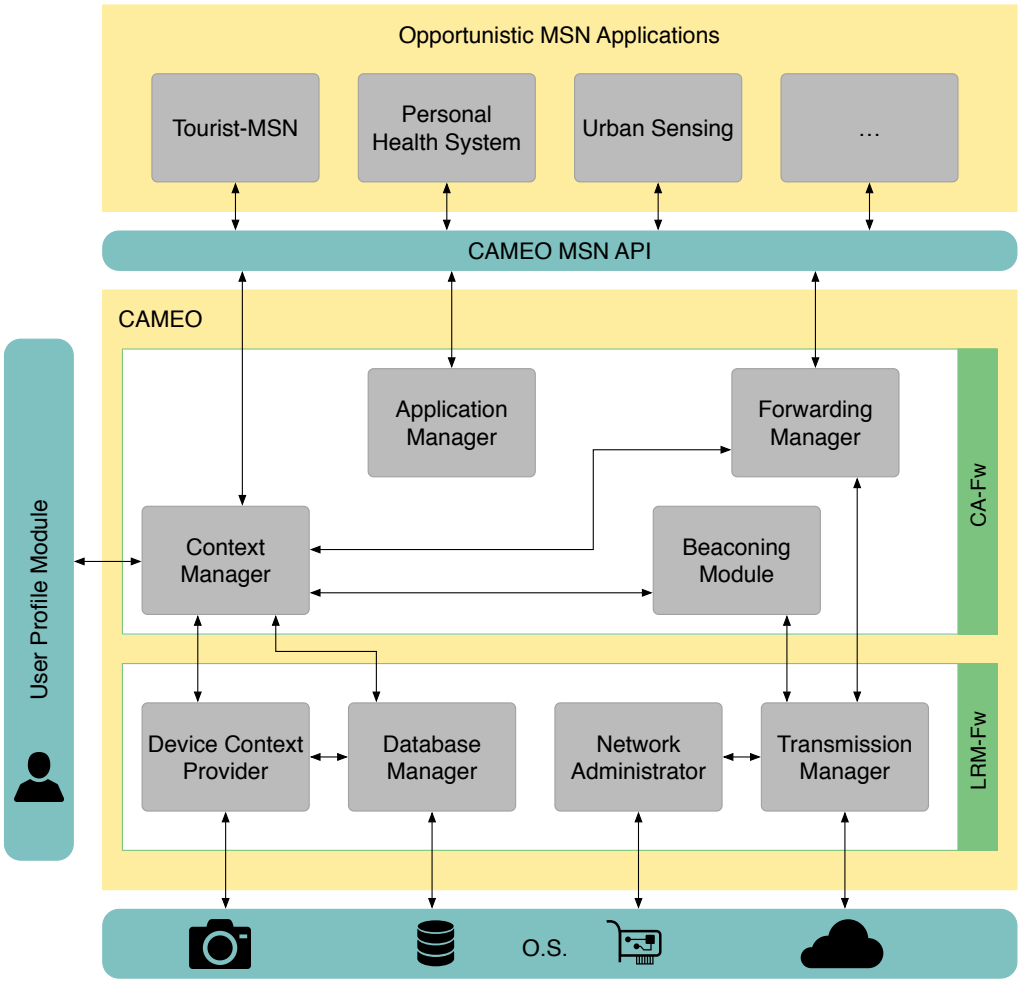


Figura 2.3: Architettura di CAMEO [6].

Lo **User Profile Module** è un modulo esterno con il quale CAMEO interagisce e ha lo scopo di mantenere tutte le informazioni che definiscono il profilo personale dell'utente. Infine, CAMEO mette a disposizione delle applicazioni MSN un insieme di API con le quali queste ultime possono interagire con il middleware.

2.3.1 Local Resource Managment Framework

LRM-Fw è composto dai seguenti quattro moduli software:

Network Administrator Per il corretto funzionamento delle applicazioni MSN, CAMEO permette ai dispositivi mobili di sfruttare tutte le occasioni di comunicazione e scambio dati attraverso contatti di tipo opportunistico. A questo scopo il Network Administrator interagisce con tutte le interfacce di comunicazione wireless a disposizione (WiFi ad-hoc, WiFi in modalità infrastruttura, Bluetooth, etc...) scegliendo sempre il miglior mezzo di comunicazione sotto specifiche condizioni. Si occupa anche di notificare le altre componenti di CAMEO interessate (principalmente il Transmission Manager) allo stato della connessione tra il nodo locale e altri dispositivi vicini (ad esempio, la qualità del collegamento WiFi oppure lo stato on/off dell'antenna Bluetooth).

Transmission Manager Dopo che il Network Manager ha selezionato l'interfaccia wireless da utilizzare per trasmettere uno specifico messaggio, il Transmission Manager si occupa di stabilire il canale di comunicazione tra i nodi sorgente e destinazione utilizzando le primitive di comunicazione standard (socket, protocolli TCP/UDP e relativi parametri). Riceve, inoltre, le notifiche da parte del Network Manager in caso di errori inerenti al link di comunicazione o disconnessioni durante la trasmissione di un messaggio.

Database Manager Responsabile delle interazioni tra l'LRM-Fw e un database SQL che si occupa di mantenere tutte le informazioni relative al well-being context di CAMEO.

Device Context Provider Si occupa di aggregare dati derivanti da diverse componenti interne del dispositivo: sensori embedded, capacità della memoria di massa, stato della batteria, consumo delle risorse, etc... Le specifiche di questi dati e i relativi parametri (come la frequenza di lettura del segnale GPS e dell'accelerometro oppure la definizione di un valore di threshold riguardo all'utilizzo della CPU) vengono definiti attraverso l'interazione con il CA-Fw, seguendo le direttive imposte dal livello applicazione o da altri moduli interni. In aggiunta, tale

modulo è in grado di gestire i dati collezionati da sorgenti eterogenee (sia interne che esterne al dispositivo mobile) così da supportare anche i servizi di sensing opportunistico e partecipativo esterni. Per garantire l'interoperabilità di CAMEO con tali sorgenti, i dati derivanti dai sensori vengono gestiti utilizzando lo standard *Sensor Mobile Enablement (SME)*[7].

2.3.2 Context-Aware Framework

CA-Fw rappresenta il fulcro di CAMEO essendo responsabile della raccolta, gestione ed elaborazione di tutte le informazioni di contesto (locale, esterno e sociale) e dello sviluppo di servizi *context-* e *social-aware* interni al middleware stesso. Il framework è formato dalle seguenti componenti:

Beaconing Module Implementa lo scambio periodico delle informazioni di contesto tra due nodi reciprocamente distanti un hop per mezzo di speciali messaggi, detti *beacon*. Questa procedura permette a CAMEO di scoprire nuovi nodi "vicini" appartenenti alla *comunità fisica* corrente e di mantenere aggiornato il *social context* del nodo locale. Solo un sottoinsieme del *local context* viene scambiato durante i contatti opportunistici. Nello specifico, le informazioni di contesto riguardanti il dispositivo non vengono scambiate a causa della loro natura. Esse infatti rappresentano delle misurazioni interne relative alle risorse locali con limitata validità temporale e vengono utilizzate solo per valutare la fattibilità di specifiche operazioni. Ad esempio, quando il nodo locale riceve una richiesta di download di un particolare contenuto da un nodo vicino, prima di accettare (e gestire tale richiesta) controlla lo stato delle proprie risorse interne come il livello della batteria. Per questo motivo e per evitare di trasmettere gran quantità di dati inutilmente, solo un sottoinsieme delle informazioni disponibili sul nodo locale vengono incluse nel *beacon*.

Forwarding Manager È il modulo responsabile della comunicazione end-to-end; in particolare è strutturato per implementare protocolli di forwarding ottimizzati per reti opportunistiche con lo scopo di inoltrare con successo i messaggi

verso le destinazioni anche in caso di connettività intermittente come, ad esempio, il protocollo HiBOp[14].

Application Manager Costituisce il canale di comunicazione tra ciascuna applicazione MSN presente nel livello superiore e CAMEO tramite le MSN API.

Context Manager È il modulo adibito alla gestione delle informazioni di contesto locale ed esterno e del loro storico. Per il mantenimento di tali informazioni si occupa di interagire con il Database Manager (lettura/scrittura delle informazioni di contesto da/sul database). In questo modulo vengono inoltre implementati algoritmi e procedure adibite al *context reasoning* e relativi adattamenti del middleware.

Allo scopo di permettere un accesso efficiente e affidabile alle informazioni di contesto a run-time, il Context Manager implementa quattro diverse strutture dati che parzialmente riflettono il contenuto del database:

- **well-being Local Context (wbLC):** contiene le informazioni relative al *local context* così organizzate: *User Profile* derivato dalle interazioni tra il Context Manager e lo User Profile Module; *Application/Service Context* specificato da ciascuna applicazione presente al livello superiore e ciascun servizio interno; *Device Context* derivato dalle interazioni tra il Context Manager e il Device Context Provider (informazioni relative al dispositivo come, ad esempio, dati dei sensori e risorse disponibili).

- **External Context (EC):** contiene i riferimenti ai contenuti disponibili e facenti parte dell'external context (dati relativi all'inquinamento ambientale, dati su condizioni meteo, etc...). In questo modo, l'EC mantiene una lista dei servizi di sensing disponibili sul nodo locale e i relativi dati presenti nel database.

- **current community Social Context (ccSC):** contiene la lista dei nodi vicini (a un hop di distanza) al nodo locale e le loro informazioni del local context disseminate tramite lo scambio periodico di messaggi di beaconing.

- **historical Social Context (hSC):** contiene la lista delle comunità precedentemente visitate dal nodo locale a cui è associato il timestamp relativo all'ultima

visita e un contatore che mantiene il numero di visite avvenute in un periodo temporale predefinito. In aggiunta, per ogni comunità mantenuta nel hSC, viene salvata anche una lista di nodi incontrati e le relative informazioni di contesto.

2.3.3 CAMEO API

Le API forniscono completo accesso alle funzionalità context- e social-aware di CAMEO per lo sviluppo di Mobile Social Network. Dal momento che la comunicazione tra il middleware e le applicazioni è sostanzialmente bidirezionale, le API possono essere divise in due sottogruppi: uno per permettere alle applicazioni di effettuare richieste al middleware, e uno per la gestione delle notifiche di CAMEO verso il livello superiore (es. messaggi, eventi, errori, etc...). Nel seguito viene riportata una descrizione ad alto livello delle possibili interazioni tra le applicazione MSN e CAMEO; le specifiche dettagliate possono essere trovate nella Sezione 2 di [5].

Registrazione Ogni applicazione deve registrarsi in CAMEO per accedere alle funzionalità offerte dal middleware. Durante la fase di registrazione viene assegnato all'applicazione un identificativo univoco e impostata l'interfaccia di callback. Nel caso in cui un'applicazione sia interessata a un servizio (interno o esterno) offerto da CAMEO (utilizzo dei sensori embedded o accesso a un servizio di sensing remoto), essa dovrà specificare il tipo di servizi a cui è interessata durante questa fase, in modo tale che CAMEO possa successivamente soddisfare la richiesta.

Specifica dell'Application Context Ciascuna applicazione deve specificare l'insieme di informazioni di contesto rilevanti per la propria esecuzione in modo tale che possa essere correttamente servita dal Context Manager di CAMEO. Tali informazioni caratterizzano le applicazioni in esecuzione sul nodo locale e vengono disseminate in rete attraverso i messaggi di beaconing.

Invio/ricezione di messaggi Le applicazioni possono inviare/ricevere messaggi tramite la comunicazione peer-to-peer e vengono notificate dal middleware in caso di errore durante l'invio.

Le notifiche di CAMEO verso le applicazioni consistono in chiamate a metodi di callback definite nelle interfacce impostate durante la procedura di registrazione. Per gestire diverse applicazioni in maniera concorrente, CAMEO assegna a ciascuna interfaccia una porta logica per la comunicazione. Una particolare porta viene utilizzata dal Context Manager per lo scambio delle informazioni di contesto via rete e le interazioni con altri moduli interni di CAMEO.

Le notifiche inviate dal middleware possono riguardare i seguenti eventi:

Nuovo contenuto Ogni volta che CAMEO rileva un nuovo contenuto da un altro nodo informa le applicazioni che possono essere interessate al dato.

Nuovo servizio Quando CAMEO scopre un nuovo servizio disponibile su un nodo remoto (incluso sensor web service o dispositivi esterni di rilevamento), notifica le applicazioni interessate a tale servizio. Quando si tratta di un servizio di sensing, CAMEO informa le applicazioni interessate inviando una descrizione del servizio e le funzionalità dei sensori coinvolti.

Nuovo vicino Quando CAMEO identifica la comparsa o l'uscita di un nuovo vicino a un hop di distanza.

Nuova comunità Quando CAMEO identifica un cambio di comunità fisica secondo l'algoritmo di community detection presente nel Context Manager (Sezione 3.2 di [6]).

Come sarà più chiaro nei prossimi capitoli, durante il lavoro di tesi è stato studiato un metodo per estendere le capacità di context reasoning del Context Manger di CAMEO. In tale modulo verrà, inoltre, implementato l'algoritmo di reasoning qui definito, PLIERS, allo scopo di ottimizzare la distribuzione dei contenuti su rete opportunistica.

Capitolo 3

Context Reasoning

Una delle principali sfide dei sistemi mobili e pervasivi è quella di dotare le applicazioni di comportamenti proattivi che, a seconda degli stimoli ricevuti dall'esterno e reagendo automaticamente a quelli più rilevanti, elaborano in maniera intelligente le informazioni di contesto inerenti sia al dispositivo che all'utente. In letteratura, questo tipo di meccanismi prende il nome di *context-reasoning*. Una definizione più precisa di context-reasoning potrebbe essere quella di dedurre nuove e rilevanti informazioni ad uso delle applicazioni e degli utenti, derivanti da varie sorgenti di dati di contesto. Facendo riferimento alla natura gerarchica delle informazioni di contesto, i dati "raw" derivanti dai sensori (*contesto di basso livello*) vengono quindi elaborati e astratti (*contesto di alto livello*) per identificare situazioni e cambiamenti del mondo reale. Per esempio, i segnali GPS possono essere mappati in locazioni astratte come *CASA*, *LAVORO*, *PALESTRA*, etc...

La maggior parte dei metodi di context-reasoning derivano principalmente dalle aree dell'intelligenza artificale (*artificial intelligence*) e apprendimento automatico (*machine learning*). In accordo con [52], è possibile classificare le tecniche di context-reasoning nelle seguenti categorie:

- *Supervised learning*: per queste tecniche è necessario avere a disposizione un *training set*, ovvero un insieme di esempi in cui ciascun dato è associato all'etichetta (*label*) ritenuta corretta. Basandosi sugli esempi contenuti nel training set viene quindi derivata una funzione per predire la label da as-

sociare a dati sconosciuti. Queste tecniche vengono tipicamente impiegate per riconoscere pattern di attività note in base ai dati provenienti dai sensori (*activity recognition*). Rientrano in questa categoria tecniche come: *Decision tree*, *Bayesian Networks*, *Artificial neural networks* e *Support vector machine (SVM)*.

- *Unsupervised learning*: queste tecniche sono in grado di riconoscere strutture e relazioni intrinseche nei dati. In assenza di un training set, non c'è nessun valore di errore o segnale di reward da valutare come potenziale soluzione. Tecniche di *clustering* come *K-Nearest Neighbour* vengono impiegate per elaborare dati di contesto di basso livello come la lettura dei sensori o per operazioni di più alto livello come la localizzazione indoor e outdoor.

- *Rule*: questa categoria rappresenta i metodi di reasoning più semplici ma anche quelli più utilizzati secondo l'analisi svolta in [39]. In questo caso il reasoning consiste nel valutare un insieme di regole espresse nella forma *IF-THEN-ELSE*; l'azione da eseguire viene determinata in base a quale regola (o regole) viene attivata dai dati in input. Queste tecniche permettono di astrarre informazioni di alto livello a partire da dati di contesto più specifici. In letteratura è possibile trovare diversi progetti in cui viene applicato un approccio *rule-based* combinato con le ontologie ([32], [66], [37]). Le tecniche basate su regole rappresentano il modo più semplice per simulare il processo di ragionamento umano in una macchina.

- *Logica fuzzy*: In contrasto con la "logica tradizionale" in cui una data proposizione può assumere solo due valori (*0* o *1*, *VERO* o *FALSO*), in quella fuzzy può assumere valori compresi nell'intervallo [0,1] indicando il *grado di verità* (o *valore di appartenenza*) della proposizione rispetto a un'opportuna funzione di appartenenza (*membership function*). L'utilizzo della logica fuzzy permette di rappresentare in maniera più coerente eventi o scenari del mondo reale utilizzando definizioni in linguaggio naturale (per indicare una temperatura: *leggermente caldo* o *abbastanza freddo*) al posto di va-

lori numerici (sempre per la temperatura: 10 °C). In altre parole, permette di rappresentare nozioni imprecise come *alto*,*corto*,*scuro* tipiche del mondo reale e che possono risultare critiche nell'elaborazione di informazioni di contesto. Tipicamente l'approccio fuzzy viene utilizzato per descrivere contesti soggettivi, effettuare la fusione di dati provenienti da più sorgenti (*multi-sensor fusion*) o risolvere potenziali conflitti tra dati di contesto.

- *Ontology based*: Data un'ontologia di riferimento, è possibile utilizzare le logiche descrittive per valutare il verificarsi di un evento o astrarre dati di contesto di più alto livello. Il principale vantaggio di queste tecniche è che si integrano bene con un context model ontologico. In contrasto, uno svantaggio del reasoning ontologico è rappresentato dall'incapacità di trovare informazioni mancanti nei dati o di elaborare informazioni ambigue; per questo motivo è necessario utilizzare altre tecniche (come quelle rule-based) a supporto del reasoning ontologico, in grado di generare ulteriori informazioni dai dati di contesto di basso livello. Tali approcci non risultano comunque molto accurati in presenza di domini caratterizzati da un alto livello di dinamismo e incertezza. Il reasoning ontologico è stato utilizzato in diversi ambiti come, ad esempio, per activity recognition [54], approcci ibridi di reasoning [54] o event detection [58].

- *Probabilistic logic*: Queste tecniche permettono di effettuare decisioni basandosi sulla probabilità associata a ciascun evento possibile del problema che si sta cercando di trattare. Vengono tipicamente impiegate per combinare assieme dati provenienti da diverse sorgenti o per risolvere conflitti tra contesti. Ad esempio, il metodo *Dempster-Shafer* permette di combinare differenti "fatti" (*evidence*) per calcolare la probabilità che si verifichi un evento. Questo metodo viene comunemente utilizzato per sensor data fusion per problemi di activity recognition (in [50] e [49] è stato impiegato per riconoscere quando è in corso una riunione in una particolare stanza). Un'altra tecnica probabilistica è rappresentata dagli *Hidden Markov Model* [11], i quali permettono di rappresentare un evento sulla base di "fatti" os-

servabili senza però essere in possesso di informazioni relative al verificarsi del particolare evento. Per esempio, avendo a disposizione i dati GPS è possibile inferire informazioni di alto livello come la possibile destinazione dell'utente o il mezzo di trasporto utilizzato.

3.1 Context Reasoning in CAMEO

Al momento della scrittura di questa tesi, in CAMEO sono stati implementati principalmente due meccanismi di reasoning: un algoritmo che potrebbe essere definito come *situation recognition* e uno di *community detection*. Il primo si occupa di identificare due insiemi di "situazioni", locali e sociali, basandosi rispettivamente su informazioni del local e social context, con lo scopo di notificare le applicazioni MSN riguardo a nuovi eventi di possibile loro interesse e adattare di conseguenza le componenti interne del middleware. Possono essere identificate diverse situazioni locali che dipendono principalmente dallo stato interno delle componenti del sistema (es. la disponibilità di nuove risorse) o da eventi relativi alle applicazioni che si interfacciano con CAMEO (registrazione di una nuova applicazione o chiusura di una di esse). Le situazioni sociali, invece, riguardano principalmente eventi generati dalla mobilità dell'utente e le sue relazioni sociali (nuovo vicino incontrato/perso, nuovi contenuti disponibili, nuovi servizi remoti, etc...). L'algoritmo di community detection si basa su uno dei concetti fondanti delle reti opportunistiche: il *traveler node* che fisicamente si sposta da una comunità di nodi all'altra. Attraverso lo scambio di informazioni di contesto tramite l'invio di messaggi di beacon con i nodi vicini, CAMEO è in grado di identificare la *current community* a cui appartiene il nodo locale sulla base dell'*home community* dichiarata dalla maggioranza dei suoi vicini.

In base a quanto esposto nella parte introduttiva del capitolo, dovrebbe essere chiaro come ciascuna tecnica di context reasoning sia caratterizzata da pregi e difetti, ed anche il motivo per il quale non esista, al momento, nessun approccio in grado di ottenere risultati perfetti. Inoltre, a causa della complessità del mondo reale, realizzare un'unica soluzione di reasoning che operi in differenti

contesti applicativi di natura diversa, risulta essere di difficile realizzazione, se non impossibile.

Più plausibile è, invece, la realizzazione di una soluzione modulare, in cui ciascuna componente sia specializzata nel reasoning relativo a un particolare problema, implementando la metodologia ritenuta più adatta allo scopo. Ogni modulo si focalizzerebbe così sull'elaborazione delle sole informazioni di contesto relative al proprio dominio e al livello di astrazione richiesto.

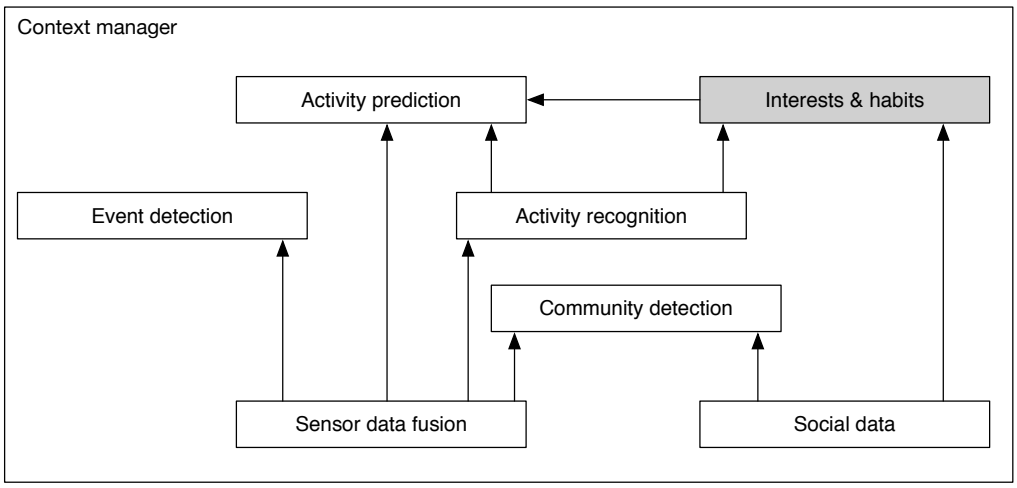


Figura 3.1: Moduli di reasoning per CAMEO.

Per dotare CAMEO di maggiori capacità di context-reasoning, in questo lavoro di tesi viene quindi proposta una soluzione modulare come in Figura 3.1. I vari moduli vengono organizzati in maniera gerarchica in base al livello di astrazione attribuito alle informazioni di contesto da essi gestite. Alla base si trovano i moduli che si occupano di elaborare le informazioni (risolvere conflitti, controllo della qualità dei dati, etc...) derivanti direttamente dai sensori e da fonti esterne. Le componenti superiori cercano invece di astrarre maggiormente le informazioni elaborate dai moduli precedenti. Ad esempio il modulo *Activity recognition*, potrebbe essere interessato a monitorare la localizzazione dell'utente e i dati provenienti da un cardiofrequenzimetro in modo tale da determinare se l'utente stia camminando o correndo, utilizzando un algoritmo di machine learning come SVM. Il modulo di *Activity prediction* potrebbe, invece, predire le

future attività dell'utente utilizzando un approccio probabilistico come la tecnica *Dempster-Shafer* inferendo su informazioni quali l'attività corrente e le sue abitudini.

Per quanto riguarda la componente evidenziata in Figura 3.1, *Interests & habits*, essa si riferisce a un modulo il cui scopo è quello di identificare gli interessi e le abitudini dell'utente basandosi su varie informazioni di contesto. Questi dati risultano utili in diversi ambiti applicativi, ma ricoprono un ruolo ancor più determinante se consideriamo il problema della distribuzione di contenuti all'interno di una rete opportunistica. Come dimostrato in [14] e lavori simili, l'efficacia degli algoritmi di routing in ambiente opportunistico aumenta notevolmente se vengono tenute in considerazione anche le informazioni di contesto e, più precisamente, gli interessi degli utenti. In questo scenario è importante determinare l'utilità ed il livello di interesse di un contenuto o di una risorsa disponibile sulla rete opportunistica per l'utente locale. Il lavoro di tesi si è quindi focalizzato nel definire un algoritmo di reasoning che, sfruttando dati di contesto relativi a contenuti e/o risorse disponibili, e gli interessi di uno o più utenti, sia in grado di calcolare l'utilità dei contenuti (o risorse) per il nodo locale, in base ad una metrica di "interesse". In una rete opportunistica un nodo non ha mai la visione completa delle informazioni che circolano in tutta la rete, ma soltanto quella relativa alle proprie informazioni e quelle dei suoi vicini. L'algoritmo dovrà quindi essere in grado di prendere decisioni analizzando un set di informazioni parziali che rispecchiano la natura frammentaria della rete opportunistica.

Capitolo 4

Distribuzione di contenuti basata sul contesto

Per quanto accennato nel Capitolo 3, lo scopo principale di questo lavoro di tesi è la realizzazione di una parte del modulo che in Figura 3.1 viene chiamato *Interests & habits*. Nella fattispecie ci si è posti il problema di come rappresentare gli interessi dell'utente e di come inferire su essi allo scopo di ottimizzare la distribuzione di contenuti nella rete opportunistica.

Lo scenario in cui opera CAMEO impone dei vincoli che sono stati tenuti in considerazione nel definire la soluzione proposta. Innanzitutto CAMEO è un middleware pensato e realizzato per essere eseguito su dispositivi mobili dotati di limitate risorse computazionali. Seppur alcuni dispositivi moderni, come gli smartphone o i tablet di fascia medio-alta, siano dotati di CPU con quattro o più core e abbiano a disposizione qualche Gigabyte di memoria RAM, i Sistemi Operativi limitano alle applicazioni l'utilizzo di tali risorse per questioni sia di sicurezza che di prestazioni dell'intero sistema. Ad esempio, il Sistema Operativo Android permette alle applicazioni di utilizzare solo qualche centinaia di Megabyte di memoria (la dimensione esatta varia a seconda del dispositivo) e, pur permettendo la loro esecuzione in background, potrebbe terminarle in qualsiasi momento (e senza preavviso) nel caso in cui applicazioni o processi con maggior priorità necessitino di ulteriori risorse. In secondo luogo, CAMEO sfrutta

le opportunità offerte dai contatti opportunistici come principale metodo di comunicazione tra i dispositivi. La frequenza e la durata di tali contatti dipendono fortemente dalla mobilità dell'utente finale. Per questi motivi, nel definire la soluzione proposta, è stata dedicata particolare attenzione a un uso efficiente della memoria e al tempo computazionale necessario.

Un metodo per identificare gli interessi dell'utente consiste nell'osservare le caratteristiche dei contenuti da esso generati o consultati. Dunque, se l'utente visiona spesso contenuti (pagine web, feed rss, etc...) inerenti ad un particolare argomento, è naturale supporre che sia interessato ad esso. Considerando, ad esempio, due generici utenti U_1 e U_2, i quali sono interessati rispettivamente agli argomenti $\{I_1, I_2\}$ e $\{I_1, I_2, I_3\}$, è logico supporre che I_3 sia legato, in qualche maniera, agli altri due. Per questo motivo, sempre nell'ottica dei contatti opportunistici, nel momento in cui i due utenti si incontrano, il sistema dovrebbe essere in grado di identificare la relazione che possibilmente esiste tra gli argomenti e suggerire (se presenti) i contenuti associati a I_3. Il sistema che si vuole realizzare dovrà quindi essere in grado di suggerire all'utente nuovi contenuti presenti in rete sulla base dei suoi interessi e relazioni sociali. Data la natura dinamica degli interessi dell'utente, la soluzione proposta dovrà essere inoltre adattabile ai cambiamenti di interessi che possono avvenire nel tempo.

In questo capitolo viene innanzitutto presentato il metodo scelto per descrivere gli interessi dell'utente esaminandone vantaggi e possibili svantaggi. Vengono poi riportati i principali algoritmi in ambito Web presenti in letteratura, in grado di consigliare dei contenuti in base ad azioni compiute in passato dall'utente. Di quelle presentate, verranno anche analizzate nel dettaglio le soluzioni che sono state ritenute le più promettenti per lo scopo prefissato.

4.1 Rappresentazione del contesto tramite tag

In CAMEO i dati che vengono generati dalle applicazioni o dai sensori vengono associati a *tag*. Questi tag sono utili sia ad identificare il tipo di contenuto (formato, tipologia di dato multimediale, etc...), sia per dare un significato semantico ai

dati. Questi *metadati* (intesi come "dati riguardanti dati") possono essere generati automaticamente dalle applicazioni oppure creati dagli stessi utenti. L'utilizzo e la generazione di contenuti (e la loro associazione a tag), possono essere considerati come indicatori degli interessi e delle preferenze dell'utente e, pertanto, risultano utili a descrivere il suo contesto.

La pratica di permettere agli utenti di assegnare liberamente dei tag alle risorse dà luogo a quella che in letteratura viene indicata come *folksonomia* [43]. Il termine folksonomia deriva dall'Inglese *folksonomy*, la quale è una combinazione delle parole *"folk"* (popolo, gente) e *"taxonomy"* (tassonomia); può essere quindi intesa come una tassonomia creata da chi la utilizza in base a criteri individuali. Per capire l'importanza che ricoprono oggigiorno le folksonomie, basti pensare ai principali social network (Facebook, Twitter, Instagram, Flickr, etc...) e alla loro presenza nella vita quotidiana. Questi sistemi permettono ai propri utenti di condividere contenuti multimediali (testo, video, foto, etc...) a cui possono essere associati uno o più tag.

Un importante aspetto delle folksonomie consiste nel fatto che non esiste alcuna gerarchia o altro tipo di relazione tra i termini che le costituiscono. Questa caratteristica è in contrasto con le tradizionali tassonomie e ontologie, in cui esistono invece diverse forme di relazioni esplicite tra i termini (generalizzazione, specializzazione e correlazione). Diversamente, le folksonomie sono dei semplici insiemi di termini creati e utilizzati dagli utenti per dare un significato semantico ai dati.

Probabilmente il punto di forza principale delle folksonomie è rappresentato dal fatto che i tag non riflettono il vocabolario di un particolare sistema di classificazione e organizzazione dei contenuti (il quale richiederebbe l'ausilio di esperti per selezionare le parole chiave più appropriate), ma il linguaggio e il vocabolario dell'utente finale. In tal senso, le folksonomie hanno la capacità quindi di adattarsi velocemente ai cambiamenti del vocabolario degli utenti e alle loro necessità. Inoltre i costi complessivi per gli utenti del sistema, in termini di tempo per descrivere un contenuto, sono nettamente inferiori rispetto a un sistema che fa affidamento su un meccanismo di classificazione e categorizzazione gerarchico.

I problemi legati all'utilizzo di un incontrollato vocabolario di termini implicano possibili limitazioni e "debolezze" intrinseche nelle folksonomie. Dal momento che non esistono vincoli semantici nell'utilizzo dei tag, uno dei problemi principali delle folksonomie consiste nell'ambiguità dei termini utilizzati dagli utenti a cui viene associato un significato diverso a seconda dei casi. Ad esempio, in [43] viene riportato un caso emblematico di questa problematica in Delicious, un social network che permette ai propri utenti di condivedere link di pagine web associando ad essi dei tag in modo tale da descrivere il loro contenuto. Alcuni link a cui è stato associato il tag "filtering" sono i seguenti:

- Last.FM - Your personal music network - Personalized online radio station
- InfoWorld: Collaborative knowledge gardening
- Wired 12.10: The Long Tail
- Oh My God It Burns! "Practical Applications of the Philosopher's stone. For drunks. Brita filter makes bad vodka into good vodka."
- Introduction to Bayesian Filtering

Come è possibile notare, questi risultati riguardano tutti, in un certo qual modo, la parola "filtering", ma con significati molto diversi tra loro: utilizzare filtri per purificare la vodka è un argomento molto diverso da un algoritmo di apprendimento statistico!

In mancanza di esplicite relazioni e dipendenze tra tag, anche l'uso di sinonimi ("mac" e "machintosh" per indicare prodotti Apple) o plurali ("flower" e "flowers") può rappresentare un problema dal punto di vista dell'analisi e utilizzo delle folksonomie.

Infine, a seguito dell'introduzione nei social network delle folksonomie, la pratica di usare tag poco significativi per la descrizione di un contenuto, ma utili solamente ad aumentare la propria visibilità e ricevere feedback positivi dagli altri utenti (tramite, ad esempio, i "like" in Facebook o i "retweet" in Twitter), + notevolmente aumentata nel tempo. In accordo con [43], questo comportamento

potrebbe essere riconducibile al fatto che tali sistemi, in qualche maniera, esaltano la necessità (definita addirittura compulsiva in certi casi) di condividere risorse e mettere in risalto la propria individualità e il proprio ego. Questi tag rappresentano quindi una sorta di "rumore", introducendo metadati non utili o rilevanti per una corretta ricerca e descrizione dei contenuti.

4.2 Reasoning basato su tag

Come introdotto precedentemente, l'algoritmo di reasoning, definito in questo lavoro di tesi, dovrà essere in grado di valutare i contenuti presenti nella rete opportunistica e suggerire all'utente quelli che risultino più in linea con i suoi interessi. In letteratura questo tipo di sistemi prende tipicamente il nome di *recommender system*, la cui implementazione è stata impiegata con successo in ambito Web per diverse applicazioni commerciali come, ad esempio, in e-commerce [55] o siti per la vendita e fruizione di contenuti multimediali (*Amazon.com* [40], *Netflix.com* [8] e *TiVo.com* [3]).

Un *recommender system* è un sistema che, utilizzando tecniche di *knowledge discovery*, permette di consigliare agli utenti dei contenuti potenzialmente di loro interesse, basandosi sulla storia delle loro attività. Tali attività possono rappresentare, ad esempio, l'interesse espresso da un utente nei confronti di un certo tipo di contenuti o le sue interazioni con altri utenti.

Con l'avvento delle folksonomie e il loro utilizzo nei principali social network, negli ultimi anni è emersa una nuova classe di sistemi di raccomandazione: i *Tag-based recommender system*. I tag utilizzati dagli utenti per descrivere i contenuti di un sistema possono essere utilizzati come utili risorse per aumentare l'efficacia degli algoritmi di raccomandazione. I vantaggi offerti dalle folksonomie nei confronti di questi sistemi possono essere riassunti nei seguenti punti:

- Dal momento che i tag sono generati e liberamente assegnati dagli utenti, essi possono essere visti come una prima indicazione riguardo le loro personali preferenze.

- I tag possono esprimere le relazioni semantiche esistenti tra contenuti diversi così da poter valutare la qualità e accuratezza dei contenuti proposti.
- Lo stesso tag può essere utilizzato per descrivere contenuti diversi. Questa proprietà può essere utilizzata sia per identificare "comunità di utenti" con interessi simili, sia per clusterizzare i contenuti, in modo tale da migliorare la qualità delle raccomandazioni.
- Utilizzando i tag generati dagli utenti è possibile risolvere il problema di *cold-start*, tipico dei recommender system. Tale problema si riferisce alla situazione in cui, non avendo ancora raccolto abbastanza informazioni, il sistema non è in grado di effettuare accurate inferenze riguardo a un utente o un contenuto.

Formalmente, in un Tag-based Recommender System, una folksonomia può essere rappresentata in uno dei seguenti modi:

1. da tre insiemi: utenti $U = \{U_1, U_2, \ldots, U_n\}$, item $I = \{I_1, I_2, \ldots, I_m\}$ e tag $T = \{T_1, T_2, \ldots, T_k\}$. Di conseguenza ciascuna relazione binaria tra essi può essere descritta tramite matrici di adiacenza **A**, **B**, **C** rispettivamente per le relazioni user-item, item-tag e user-tag.

   Se l'utente U_i ha collezionato l'item I_j viene impostato $a_{i,j} = 1$, altrimenti $a_{i,j} = 0$; analogamente se il tag T_k è stato assegnato all'item I_j viene settato $b_{j,k} = 1$, altrimenti $b_{j,k} = 0$, infine se l'utente U_i ha adottato per qualche ragione il tag T_k allora $c_{i,k} = 1$, in caso contrario $c_{i,k} = 0$;

2. tramite una struttura ternaria in cui ciascuna relazione (u, i, t) viene rappresentata da una componente $Y = 1$ se la relazione esiste nella folksonomia, $Y = 0$ in caso contrario.

In questa sezione vengono riportati gli approcci che definiscono lo stato dell'arte per i Tag-based Recommender System. Vengono dunque esaminate dettagliatamente quelle soluzioni (i modelli network-based) ritenute più promettenti allo scopo di realizzare un algoritmo di reasoning per CAMEO orientato alla distribuzione dei contenuti basata su informazioni di contesto.

4.2.1 Modelli Network-Based

I modelli *network-based* utilizzano i grafi, principalmente bipartiti e tripartiti, come strutture dati per rappresentare le relazioni tra gli insiemi che costituiscono la folksonomia (*user-tag*, *user-item*, *item-tag* o *user-item-tag*).

In un tipico grafo bipartito sono presenti due insiemi di vertici mutuamente collegati e in cui non esistono archi che collegano elementi del medesimo insieme.

Questo tipo di sistemi di raccomandazione considera simili due elementi se hanno dei collegamenti in comune, cioè se hanno archi verso gli stessi nodi. Ad esempio, un Recommender System per brani musicali potrebbe utilizzare un grafo in cui sono presenti due tipi di nodi, "utente" e "brano", in cui esiste un arco tra due nodi solo se l'utente U_i ha ascoltato il brano B_j. Riferendosi al semplice grafo di Figura 4.1, il sistema potrebbe raccomandare all'utente $U1$ le canzoni $B4$ e $B6$ collegati all'utente $U2$. Questo perchè, avendo i due utenti dei brani in comune ($B2$ e $B3$), quelli proposti potrebbero essere simili alle canzoni precedentemente ascoltate da $U1$ (o, almeno, potrebbero essere di suo interesse).

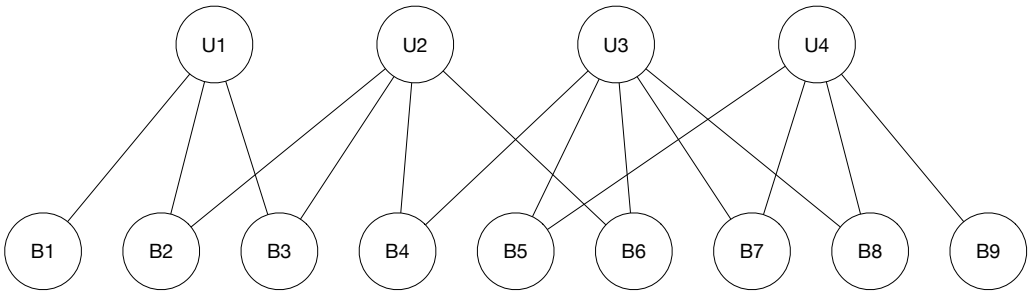


Figura 4.1: Grafo bipartito "user-brano".

Ispirandosi a queste strutture dati e applicando la teoria dei grafi nei Recommender system, in letteratura sono stati proposti due principali algoritmi: *Probability Spreading (ProbS)* e *Heat Spreading (HeatS)*.

ProbS [65, 64] Supponendo di avere un grafo bipartito *user-item* con n user e m item in cui la presenza di un arco tra l'utente U_i e l'item I_j indica una generica relazione tra i due (ad esempio il download dell'item da parte dell'utente), si vuole

calcolare *quanto* gli item non direttamente collegati all'utente U_i (*utente target*) siano potenzialmente interessanti per esso.

Il valore finale di ciascun item, f_j^p, per l'utente target U_i viene calcolato da ProbS in accordo alle seguenti regole.

Inizialmente viene assegnata una generica risorsa a ciascun item I_j in modo tale che valga 1 se $a_{ij} = 1$ e 0 altrimenti, cioè se è presente o meno un arco tra l'utente U_i e l'item I_j.

La risorsa viene divisa (*primo step di diffusione*) per il numero di utenti collegati all'item e ogni utente riceve la stessa porzione di risorsa. Successivamente, ogni utente divide la porzione di risorsa ricevuta per il numero di item ad esso collegati (*secondo step di diffusione*) e ciascun item riceve quindi la medesima porzione di risorsa.

Il valore finale di ciascun item I_j è dato dalla somma delle porzioni di risorse che vengono assegnate ad esso dopo i due step di diffusione.

Formalmente il *vettore di risorse* finale f^p (cioè l'insieme di tutti i valori f_j^p calcolati) per l'utente target U_i, è dato dalla formula:

$$f_j^p = \sum_{l=1}^{n} \sum_{s=1}^{m} \frac{a_{lj} a_{ls} a_{is}}{k(U_l) k(Is)} \qquad j = 1, 2, \ldots, m \tag{4.1}$$

dove $k(U_l) = \sum_{j=1}^{m} a_{lj}$ è il numero di item collezionati dall'utente U_l (numero di archi collegati a U_l), e $k(I_s) = \sum_{i=1}^{n} a_{is}$ è il numero di utenti interessati all'item I_s (numero di archi collegati a I_s). In questo caso il vettore di risorse finale è utilizzato per definire una classifica dei contenuti disponibili rispetto agli interessi dell'utente, selezionando però solo quelli non ancora in possesso dell'utente locale (Tabella 4.1).

In Figura 4.2a,4.2b e 4.2c è riportato un esempio di applicazione di ProbS con un grafo bipartito user-item. Per l'utente target U_1 il calcolo dei valori degli item avviene come segue:

1. vengono assegnate le risorse agli item collezionati dall'utente target, $f_{I_1} = f_{I_4} = 1$ e $f_{I_2} = f_{I_3} = f_{I_5} = 0$
2. viene eseguito il primo step di diffusione: le risorse vengono distribuite dagli item agli utenti ad essi collegati.
   $f_{U_1} = f_{I_1} \times \frac{1}{3} + f_{I_4} \times \frac{1}{2} = \frac{5}{6}$
   $f_{U_2} = f_{I_1} \times \frac{1}{3} + f_{I_2} \times 1 + f_{I_3} \times \frac{1}{3} + f_{I_4} \times \frac{1}{2} = \frac{5}{6}$
   $f_{U_3} = f_{I_1} \times \frac{1}{3} + f_{I_3} \times \frac{1}{3} = \frac{1}{3}$
   $f_{U_4} = f_{I_3} \times \frac{1}{3} + f_{I_5} \times 1 = 0$
3. infine viene eseguito il secondo step di diffusione: le risorse vengono redistribuite dagli utenti agli item ad essi collegati.
   $f^p_{I_1} = f_{U_1} \times \frac{1}{2} + f_{U_2} \times \frac{1}{4} + f_{U_3} \times \frac{1}{2} = \frac{19}{24}$
   $f^p_{I_2} = f_{U_2} \times \frac{1}{4} = \frac{5}{24}$
   $f^p_{I_3} = f_{U_2} \times \frac{1}{4} + f_{U_3} \times \frac{1}{2} + f_{U_4} \times \frac{1}{2} = \frac{3}{8}$
   $f^p_{I_4} = f_{U_1} \times \frac{1}{2} + f_{U_2} \times \frac{1}{4} = \frac{5}{8}$
   $f^p_{I_5} = f_{U_4} \times 1 = 0$

Tabella 4.1: Classifiche prodotte da ProbS e HeatS per l'esempio di Figura 4.2. Le posizioni evidenziate si riferiscono agli item già in possesso dell'utente target.

Position	Probability Spreading (ProbS)	Heat Spreading (HeatS)
1°	I1	I4
2°	I4	I1
3°	I3	I2
4°	I2	I3
5°	I5	I5

HeatS [61, 64] Il funzionamento di HeatS è simile a quello di ProbS, ma basato su regole inverse.

Durante il primo step di diffusione delle risorse, ciascuna di esse viene divisa per il numero di item collegati ad ogni utente; al secondo step, quando le porzioni di risorse vengono redistribuite dagli utenti agli item, le porzioni verranno divise per

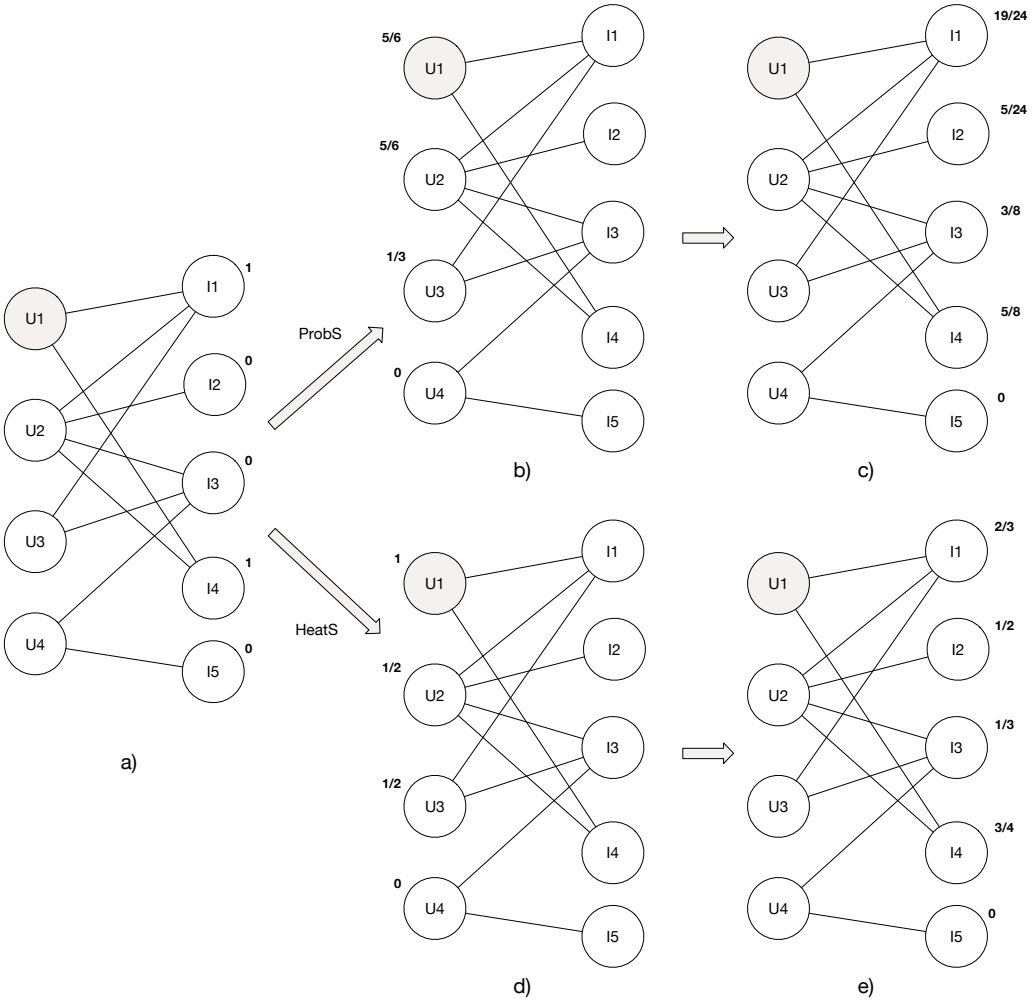


Figura 4.2: ProbS e HeatS con un grafo bipartito user-item in cui U1 è l'utente target [64].

il numero di utenti collegati a ciascun item. In altre parole, ogni volta che una risorsa (o porzione di essa) viene ridistribuita, viene divisa per il numero di archi collegati al nodo verso il quale si sta dirigendo.

In questo caso il *vettore di risorse* finale f^h, per l'utente U_i, è dato dalla formula:

$$f_j^h = \frac{1}{k(I_j)} \sum_{l=1}^{n} \sum_{s=1}^{m} \frac{a_{lj} a_{ls} a_{is}}{k(U_l)} \qquad j = 1, 2, \ldots, m \tag{4.2}$$

In Figura 4.2a,4.2d e 4.2e è riportato un esempio di applicazione di HeatS:

1. vengono assegnate le risorse agli item collezionati dall'utente target, $f_{I_1} = f_{I_4} = 1$ e $f_{I_2} = f_{I_3} = f_{I_5} = 0$
2. viene eseguito il primo step di diffusione: le risorse vengono distribuite dagli item ai corrispettivi utenti.
   $f_{U_1} = f_{I_1} \times \frac{1}{2} + f_{I_4} \times \frac{1}{2} = 1$
   $f_{U_2} = f_{I_1} \times \frac{1}{4} + f_{I_2} \times \frac{1}{4} + f_{I_3} \times \frac{1}{4} + f_{I_4} \times \frac{1}{4} = \frac{1}{2}$
   $f_{U_3} = f_{I_1} \times \frac{1}{2} + f_{I_3} \times \frac{1}{2} = \frac{1}{2}$
   $f_{U_4} = f_{I_3} \times \frac{1}{2} + f_{I_4} \times \frac{1}{2} = 0$
3. viene eseguito il secondo step di diffusione: le risorse vengono redistribuite dagli utenti agli item ad essi collegati.
   $f^p_{I_1} = f_{U_1} \times \frac{1}{3} + f_{U_2} \times \frac{1}{3} + f_{U_3} \times \frac{1}{3} = \frac{2}{3}$
   $f^p_{I_2} = f_{U_2} \times 1 = \frac{1}{2}$
   $f^p_{I_3} = f_{U_4} \times \frac{1}{3} + f_{U_3} \times \frac{1}{3} + f_{U_2} \times \frac{1}{3} = \frac{1}{3}$
   $f^p_{I_4} = f_{U_1} \times \frac{1}{2} + f_{U_2} \times \frac{1}{2} = \frac{3}{4}$
   $f^p_{I_5} = f_{U_4} \times 1 = 0$

Soluzione ibrida [42] Mentre ProbS tende a mettere in risalto gli item più popolari presenti in rete (assegnando cioè un valore maggiore a quelli collegati a più utenti), HeatS ne riduce il valore ed è incline a raccomandare quelli che hanno meno archi verso gli utenti. Questa differenza tra i due algoritmi è possibile notarla anche in Figura 4.2 e Tabella 4.1, dalle quali risulta evidente che ProbS consiglierà l'item I_3 (quello con più utenti collegati) e invece HeatS suggerirà l'item I_2.

Queste caratteristiche possono risultare limitanti in alcuni scenari. Infatti, se volessimo migliorare l'accuratezza delle raccomandazioni in base alla popolarità degli item già in possesso dell'utente target, ciascuno dei due algoritmi potrebbe mostrare un comportamento indesiderato in diverse situazioni. Ad esempio, se l'utente fosse interessato solo ad item poco popolari, ProbS sbaglierebbe consigliando item con popolarità molto più elevata; al contrario, se l'utente fosse interessato ad item particolarmente popolari, HeatS sbaglierebbe consigliando item con bassa popolarità.

Una soluzione a questo problema potrebbe essere rappresentata da un approccio di tipo ibrido. Un semplice metodo per realizzare gli algoritmi ibridi è quella di combinare linearmente i risultati di due algoritmi [16]. Se due generici approcci, X e Y, generano i valori, rispettivamente x_α e y_β, il risultato ibrido può essere ottenuto con la seguente formula:

$$(1-\lambda)\left[\frac{x_\alpha}{\max\limits_\beta x_\beta}\right]+\lambda\left[\frac{y_\alpha}{\max\limits_\beta y_\beta}\right] \tag{4.3}$$

dove l'operazione di normalizzazione è dovuta al fatto che due metodi diversi possano produrre risultati su due scale di valori molto differenti tra loro. Con il parametro $\lambda \in [0,1]$, detto *parametro di ibridazione*, è possibile calibrare l'algoritmo ibrido in favore delle caratteristiche di un metodo piuttosto che dell'altro. A discapito della sua semplicità, questo approccio presenta lo svantaggio di richiedere l'esecuzione di entrambi gli algoritmi prima di restituire un risultato, raddoppiando così i costi computazionali.

Una soluzione ibrida ProbS+HeatS molto più elegante viene proposta in [42] con la seguente formula:

$$f_j^h = \sum_{l=1}^{n}\sum_{s=1}^{m}\frac{a_{lj}a_{ls}a_{is}}{k(U_l)}\frac{1}{k(I_s)^\lambda k(I_j)^{1-\lambda}} \qquad j=1,2,\ldots,m \tag{4.4}$$

dove con $\lambda = 0$ si ottiene l'algoritmo HeatS, e con $\lambda = 1$ si ottiene invece ProbS. Al contrario della Formula 4.3, questa soluzione è caratterizzata da una complessità computazionale pari a quella dei singoli ProbS e Heats, offrendo però al contempo una raccomandazione intermedia tra i due per valori di λ compresi nell'intervallo $(0,1)$.

SimRank [36] SimRank è un algoritmo per il calcolo della similarità tra due generici oggetti che si basa sull'assunzione che *"due oggetti sono simili se sono collegati a oggetti simili tra loro"*. Più precisamente, due oggetti *a* e *b* sono simili se e solo se sono collegati rispettivamente agli oggetti *c* e *d* e questi ultimi risultano già essere simili tra loro.

Il flusso computazionale dell'algoritmo può essere rappresentato da un grafo orientato G^2, in cui ciascun nodo rappresenta una coppia ordinata di vertici del grafo di partenza G con il quale sono rappresentate le relazioni tra gli oggetti. In G^2 esiste un arco tra i nodi (a,b) e (c,d) se, in G, esiste un arco tra a e c e un arco tra b e d. Il valore calcolato da SimRank per un dato nodo v in G^2, indica il grado di similarità tra i due nodi di G rappresentati dal vertice v.

Denotando con $I(v)$ e $O(v)$ rispettivamente gli insiemi degli archi entranti e uscenti del nodo v, la similarità tra due oggetti, $s(a,b) \in [0,1]$, è definita dalla seguente equazione ricorsiva:

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) \tag{4.5}$$

in cui C è una costante tra 0 e 1 e se $a = b$ allora $s(a,b) = 1$.

Supponendo che due persone, A e B, abbiano acquistato rispettivamente gli item $\{uova, crema, zucchero\}$ e $\{uova, crema, farina\}$, il grafo bipartito che rappresenta tali relazioni è riportato in Figura 4.3(a).

La similarità tra gli item e quella tra gli utenti sono nozioni che si rinforzano reciprocamente:

- due utenti sono simili se hanno acquistato item simili. Formalmente la similarità tra due utenti è la media delle similarità tra gli item che hanno acquistato:

$$s(A,B) = \frac{C_1}{|O(A)||O(B)|} \sum_{i=1}^{|O(A)|} \sum_{j=1}^{|O(B)|} s(O_i(A), O_j(B)) \tag{4.6}$$

- due item sono simili se sono stati acquistati da utenti simili. Formalmente la similarità tra due item è la media delle similarità tra gli utenti da cui sono stati acquistati:

$$s(c,d) = \frac{C_2}{|I(c)||I(d)|} \sum_{i=1}^{|I(c)|} \sum_{j=1}^{|I(d)|} s(I_i(c), I_j(d)) \tag{4.7}$$

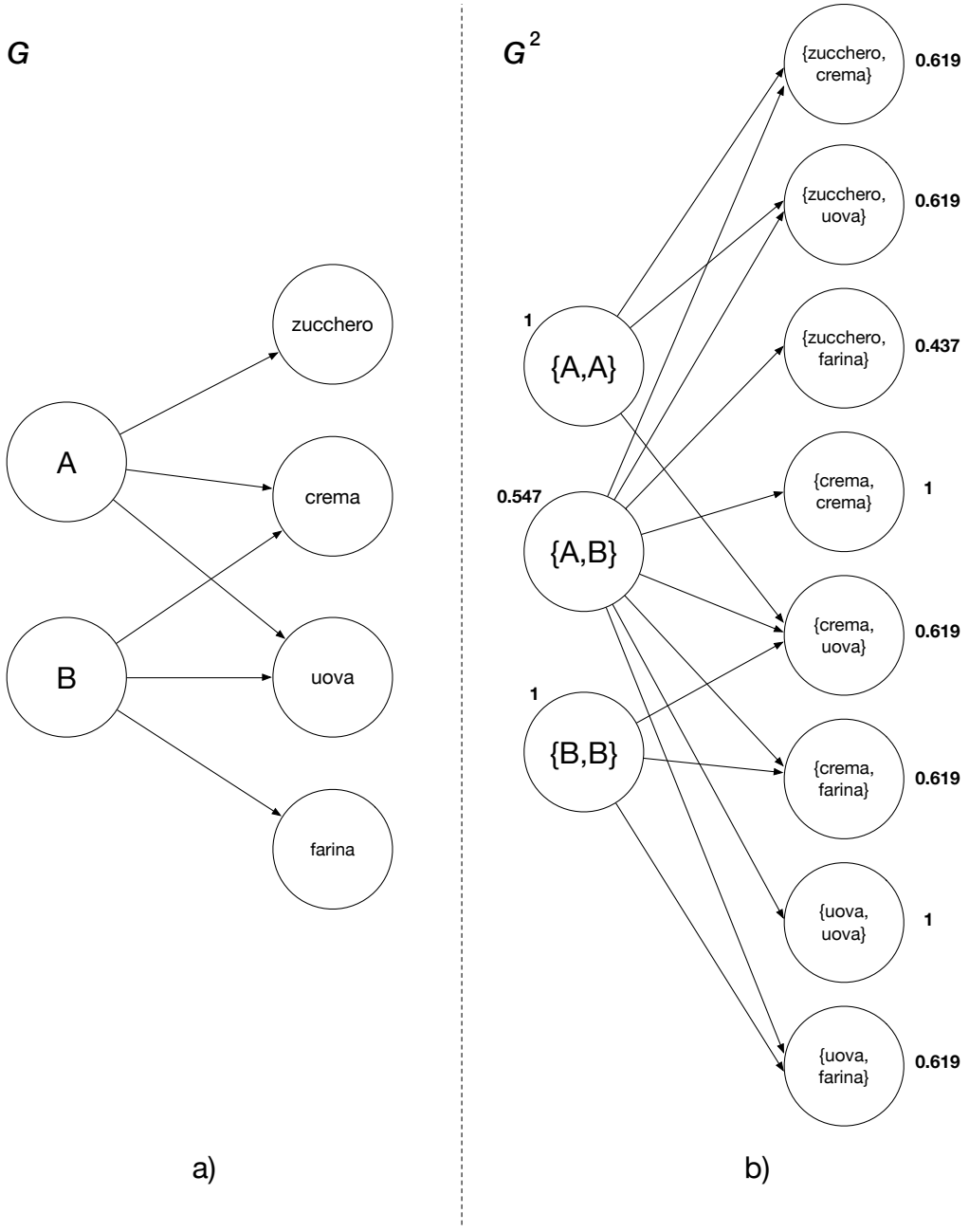


Figura 4.3: In a) il grafo degli acquisti degli utenti A e B. In b) la versione semplificata del grafo delle coppie dei nodi G^2. Gli score di SimRank in b) sono stati calcolati usando $C_1 = C_2 = 0.8$ [36].

Una soluzione alle equazioni 4.6 e 4.7 può essere ottenuta dopo un numero K di iterazioni dell'algoritmo. Se n è il numero di nodi del grafo G, per ogni iterazione k è necessario mantenere n^2 vettori $R_k(*,*)$ ciascuno di cardinalità n^2, dove $R_k(a,b)$ indica il valore di similarità tra a e b calcolato alla k-esima iterazione. Successivamente verrà calcolato $R_{k+1}(*,*)$ basandosi sui valori contenuti in $R_k(*,*)$.

L'algoritmo viene inizializzato con $R_0(*,*)$ dove ciascun $R_0(a,b)$ è dato da:

$$R_0(a,b) = \begin{cases} 0, & \text{se } a \neq \text{ b,} \\ 1, & \text{altrimenti} \end{cases} \tag{4.8}$$

Per calcolare $R_{k+1}(a,b)$ dai valori di $R_k(*,*)$ viene usata l'equazione 4.5 per ottenere:

$$R_{k+1}(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) \tag{4.9}$$

per ogni $a \neq b$ e $R_k(a,b) = 1$ per $a = b$. Quindi, all'iterazione $k+1$, la similarità $s(a,b)$ viene aggiornata usando il valore di similarità calcolato durante la precedente iterazione k in accordo con l'equazione 4.5.

In [35] viene mostrata la convergenza dell'algoritmo per $k \to \infty$ e, più precisamente, vale che $\lim_{k \to +\infty} R_k(a,b) \approx s(a,b)$. Inoltre, gli autori di SimRank riportano che in tutti i loro esperimenti hanno riscontrato una rapida convergenza dell'algoritmo, con i valori di ranking che si stabilizzavano dopo sole cinque iterazioni.

Seppur l'algoritmo sembra essere promettente, il problema di questo approccio consiste nel fatto che risulta essere troppo oneroso per lo scenario di riferimento, ovvero un contesto di reti opportunistiche in cui ogni nodo è dotato di risorse limitate. Infatti, dato un grafo G composto da n nodi, ipotizzando che il sistema si stabilizzi dopo K iterazioni e indicando la media di $|I(a)||I(b)|$ su tutte le cop-

pie (a,b) con d_2, la complessità in tempo di SimRank risulta essere $O(d_2 n^2 K)$ e un'occupazione di memoria pari a $O(n^2)$.

4.2.2 Altri approcci

In questa sezione vengono riportati altri metodi per la realizzazione di Tag-based recommender system. Seppur promettenti, tali algoritmi non risultano però essere applicabili allo scenario considerato in questo lavoro di tesi, in quanto caratterizzato da una limitata capacità computazionale dei dispositivi e vincoli di tempo stringenti dovuti alla natura dei contatti opportunistici. Non si esclude comunque la loro applicazione per altre operazioni di reasoning in cui il fattore tempo non sia così determinante.

Modelli Tensor-based I tensori possono essere descritti informalmente come delle "matrici a più dimensioni" contenenti valori arbitrari e sono tipicamente utilizzati per descrivere vari fenomeni in diversi campi della fisica. Recentemente sono stati impiegati anche per definire algoritmi per recommender system basati su tag.

In [53] viene presentato un algoritmo di apprendimento (*RTF - ranking with tensor factorization*) per cercare di predire quali tag un utente potrebbe usare per descrivere un particolare contenuto basandosi sulla sua storia passata.

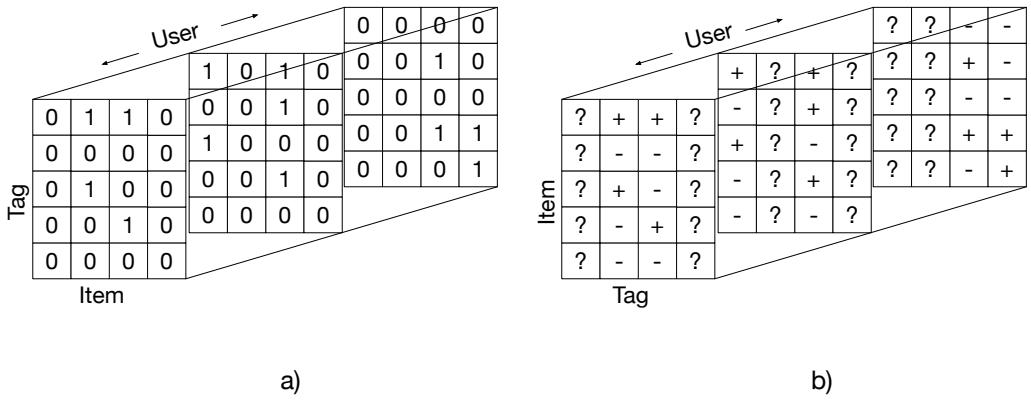


Figura 4.4: Creazione del training set con (a) lo schema 0/1 e (b) lo schema Post-based ranking [53].

Dati U, I, T rispettivamente gli insiemi degli utenti, item e tag, viene definito $S \subseteq U \times I \times T$ come l'insieme delle informazioni della storia passata degli utenti. La terna $(u,i,t) \in S$ indica che l'utente u ha utilizzato in passato il tag t per descrivere l'item i. S può quindi essere rappresentato come un cubo, le cui dimensioni sono rispettivamente: gli utenti, gli item e i tag.

Da S viene ricavato un tensore Y contenente tutti i dati del training set. Gli esempi facenti parte del training set possono essere ottenuti tramite uno dei seguenti metodi:

- *0/1 Interpretation scheme*. Ciascun esempio positivo viene codificato con *1* e gli altri con *0*. In questo caso, se una particolare terna $(utente, item, tag)$ si trova in S viene considerata come esempio positivo, altrimenti come esempio negativo. Il training set $Y^{0/1}$ è definito quindi come:

$$y^{0/1}_{(u,i,t)} = \begin{cases} 1, & \text{se } (u,i,t) \subseteq \text{ S,} \\ 0, & \text{altrimenti} \end{cases} \tag{4.10}$$

- *Post-based Ranking Interpretation Scheme*. In questo metodo gli esempi positivi e negativi vengono identificati dai soli dati osservabili in S. Nel caso in cui un utente non abbia mai descritto un item con alcun tag, viene identificato come "dato mancante" e non verrà considerato come esempio del training set come invece avviene nello schema 0/1.

In Figura 4.4 viene rappresentata graficamente l'applicazione dei due metodi per creare il dataset basandosi sulle informazioni relative a tre utenti di esempio.

A questo punto l'algoritmo si basa sulla tecnica di *Tensor Factorization* [53] per generare il tensore risultante $\widehat{Y}$, contenente le predizioni per ciascun utente. Utilizzando un metodo di *low-rank minimization* [53], le matrici U,I e T vengono ridotte nelle *feature matrix* $\widehat{U}$,$\widehat{I}$,$\widehat{T}$ caratterizzate da un rango inferiore rispetto a quelle di partenza. Il tensore viene quindi costruito moltiplicando le tre *feature matrix* $\widehat{U}$,$\widehat{I}$,$\widehat{T}$ con un tensore di dimensioni ridotte $\widehat{C}$, detto *core tensor*.

$$\widehat{Y} := \widehat{C} \times_u \widehat{U} \times_i \widehat{I} \times_t \widehat{T} \tag{4.11}$$

Dove il core tensor $\widehat{C}$ e le tre feature matrix $\widehat{U}$,$\widehat{I}$ e $\widehat{T}$ sono parametri del modello che devono essere appresi, e $\times_x$ indica il prodotto tensoriale per moltiplicare una matrice di dimensione x con un tensore.

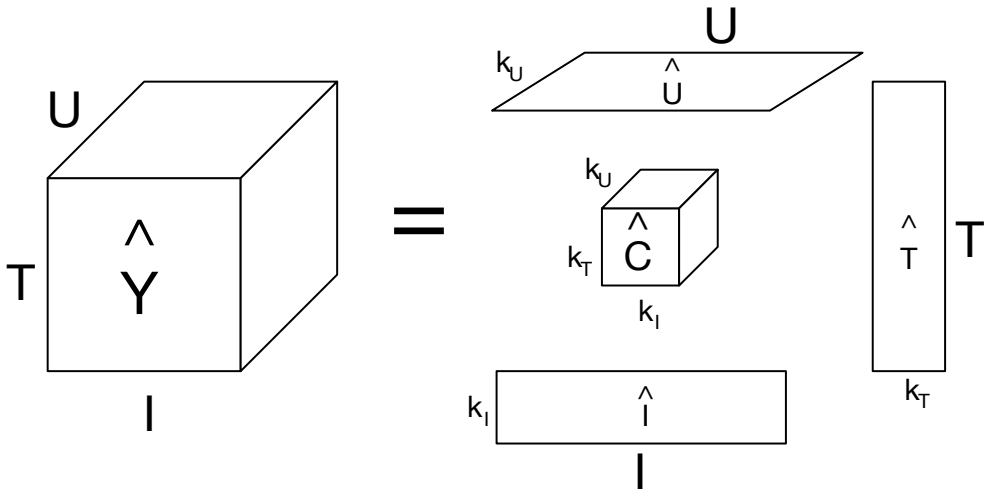


Figura 4.5: Fattorizzazione del tensore: il tensore $\widehat{Y}$ viene costruito moltiplicando tre feature matrix $\widehat{U}$,$\widehat{I}$,$\widehat{T}$ con un core tensor $\widehat{C}$ [53].

I parametri del modello hanno le seguenti dimensioni: $\widehat{C} \in \mathbb{R}^{k_U \times k_I \times k_T}$, $\widehat{U} \in \mathbb{R}^{|U| \times k_U}$, $\widehat{I} \in \mathbb{R}^{|I| \times k_I}$, $\widehat{T} \in \mathbb{R}^{|T| \times k_T}$. Il tensore risultante dalla formula 4.11 avrà quindi dimensione pari a $|U| \times |I| \times |T|$.

Date le feature matrix e il core tensor, la predizione $\widehat{y}_{u,i,t}$viene calcolata come segue:

$$\widehat{y}_{u,i,t} = \sum_{\widetilde{u}} \sum_{\widetilde{i}} \sum_{\widetilde{t}} \widehat{c}_{\widetilde{u},\widetilde{i},\widetilde{t}} \cdot \widehat{u}_{u,\widetilde{u}} \cdot \widehat{t}_{t,\widetilde{t}} \tag{4.12}$$

dove gli indici che si riferiscono alle dimensioni delle feature matrix sono segnati con il simbolo tilde ($\widetilde{}$) e i loro elementi vengono invece indicati con un "cappuccio" (accento circonflesso,$\widehat{}$). Infine, dati gli $\widehat{y}_{u,i,t}$, è possibile ottenere la lista

personalizzata degli N tag più consigliati per l'utente u e l'item i con la seguente formula:

$$Top(u,i,N) := \underset{t \in T}{\operatorname{argmax}}^{N} \widehat{y}_{u,i,t} \tag{4.13}$$

Per effettuare una raccomandazione tali sistemi richiedono quindi di fattorizzare il tensore tramite tecniche basate sulla *Singular Value Decomposition (SVD)* [26], le quali richiederebbero una complessità computazionale in tempo troppo elevata per il contesto considerato.

Modelli Topic-based Generalmente parlando, la sfida principale dei recommender system è quella di riuscire a stimare la probabilità che un utente possa essere interessato a un certo item. In letteratura sono stati proposti diversi approcci che, con l'ausilio di varie tecniche statistiche, cercano di stimare tale probabilità.

Ad esempio, in [22] viene proposto l'algoritmo *LSA (Latent Semantic Analysis)* che, basandosi su una matrice di frequenza, permette di estrarre i termini più significativi da un documento di testo in modo tale da identificare l'argomento (*topic*) soggiacente. Un'evoluzione di LSA è *PLSA (Probability Latent Semantic Analysis)* [31]. Tale algoritmo calcola la probabilità di co-occorrenza di due eventi (parole e documenti in questo caso), dando per assunto un modello latente, cioè non noto a priori, dell'argomento di appartenenza tramite la Formula 4.14.

$$P(w,d) = \sum_{z} P(z)P(d|z)P(w|z) = P(d)\sum_{z} P(z|d)P(w|z) \tag{4.14}$$

Il problema principale di tale approccio risiede nel fatto che, per avere una buona stima, la probabilità di interesse deve essere calcolata su un numero considerevole di dati, cosa che non può essere assunta nel caso di CAMEO. Il sistema dovrà essere infatti in grado di effettuare delle raccomandazioni anche in presenza di poche informazioni e fin dai primi contatti opportunistici.

Capitolo 5

PLIERS: Popularity-based item Recommender System

Ai fini di definire un algoritmo di reasoning che sia in grado di migliorare la distribuzione di contenuti su rete opportunistica per CAMEO, è stato scelto un approccio di tipo network-based (descritto nel Capitolo 4.2.1) per la bassa complessità computazionale e la facile implementazione che contraddistingue tale metodo.

ProbS e *HeatS*, così come sono stati proposti in letteratura, favoriscono gli item più popolari il primo, e quelli meno popolari il secondo. La soluzione ibrida ProbS+HeatS (qui chiamata semplicemente *"Hybrid"*), invece, permette di ottenere delle raccomandazioni intermedie, ma comunque non soddisfacenti per gli scopi prefissati. Il comportamento desiderato è infatti quello di valorizzare i contenuti con un livello di popolarità simile a quelli dell'utente considerato, in modo tale da adattare il più possibile le raccomandazioni effettuate agli interessi dell'utente.

In questo lavoro si assume che un item, ad esempio un tag, molto popolare (collegato a molti utenti), possa semanticamente riferirsi a un argomento più "generico" rispetto ad uno poco popolare che, invece, si intende descrivere un argomento più "specifico". Ad esempio, in Figura 5.1, viene riportato un grafo bipartito in cui sono rappresentati degli utenti associati ai relativi interessi (archi tra user e item). Considerando $U1$ come utente target, si vorrebbe che non venga

mai proposto l'item $I4$ poichè è troppo generico rispetto agli interessi di $U1$ ($I1$ e $I2$); al contrario, invece,considerando $U4$ come utente target, non dovrebbe essere proposto l'item $I3$ in quanto è troppo specifico rispetto a $I4$.

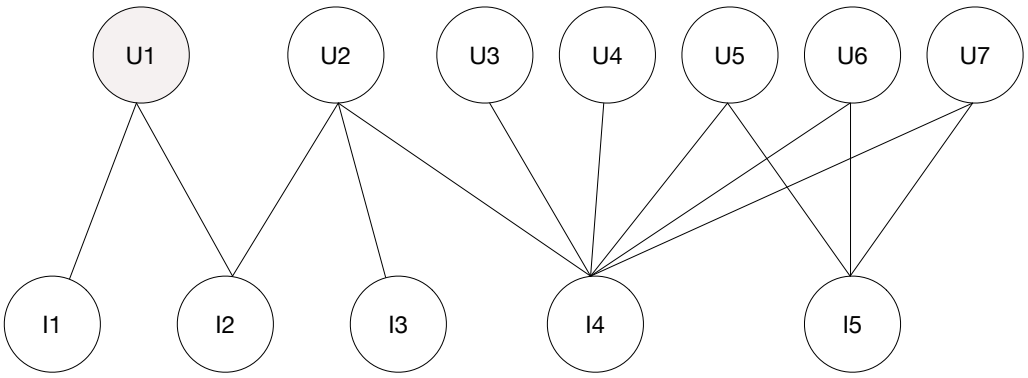


Figura 5.1: $I4$ è un item più generico rispetto agli interessi dell'utente $U1$.

In questo capitolo viene presentata la soluzione proposta, dimostrando la sua efficiacia per lo scenario di riferimento e confrontandola con ProbS, HeatS e Hybrid. Viene inoltre descritta la sua applicazione in ambiente opportunistico e il meccanismo utilizzato per rendere l'algoritmo adattivo ai contenuti che vengono condivisi in rete, in modo tale da rendere più accurate le raccomandazioni effettuate in base agli interessi dell'utente locale.

5.1 L'algoritmo

Allo scopo di privilegiare i contenuti con popolarità simile a quella degli item già in possesso dell'utente locale, viene proposto un nuovo algoritmo: *PLIERS (PopuLarity-based ItEm Recommender System)*.

PLIERS si ispira al funzionamento di ProbS, con la differenza che il valore di similarità tra due item, I_j e I_s, viene normalizzato per il numero di utenti collegati a I_j ma non connessi a I_s. Formalmente, indicando con $U(I_j)$ e $U(I_s)$ rispettivamente l'insieme degli utenti collegati all'item I_j e l'insieme di quelli collegati a I_s, il valore di normalizzazione viene indicato con $k(I_j - I_s) = \left|U(I_j) \setminus U(I_s)\right|$. In questo modo l'algoritmo è in grado di normalizzare il valore calcolato per la differenza di popolarità tra i due item esaminati.

Il *vettore di risorse finale* per l'utente target U_i, in un grafo costituito da n utenti e m item, viene quindi calcolato utilizzando la seguente formula:

$$f_j = \sum_{l=1}^{n} \sum_{s=1}^{m} \frac{a_{lj} \cdot a_{ls} \cdot a_{is}}{k(U_l) \cdot k(Is)} \cdot \frac{1}{k(I_j - I_s) + 1} \qquad j = 1, 2, \ldots, m \tag{5.1}$$

dove $k(U_l) = \sum_{j=1}^{m} a_{lj}$ è il numero di item collezionati dall'utente target U_l e $k(I_s) = \sum_{i=1}^{n} a_{is}$ è il numero di utenti interessati all'item I_s.

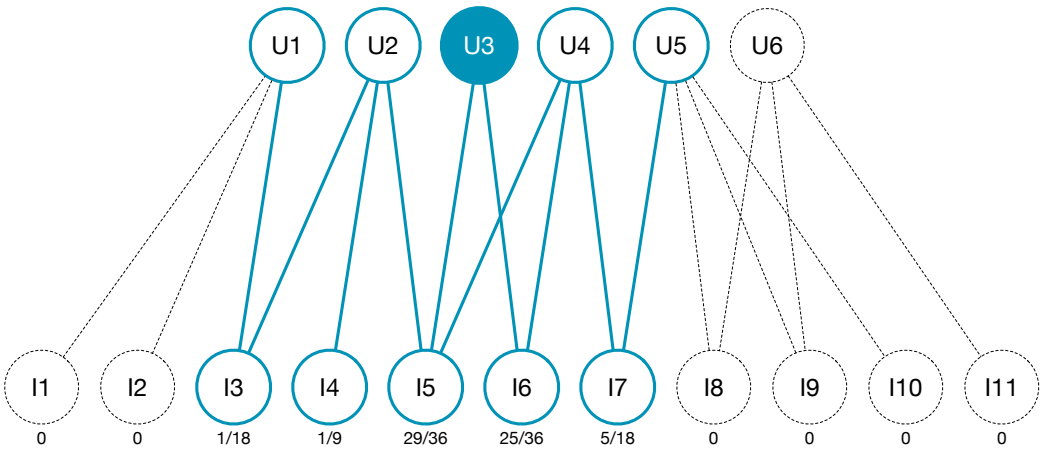


Figura 5.2: Vertici e archi essenziali per l'esecuzione di PLIERS considerando $U3$ come utente target. Gli item tratteggiati ottengono un valore nullo.

L'equazione 5.1 è stata volutamente trascritta mantenendo la forma di quelle di PlierS e HeatS in [63], in modo tale da evidenziare le relazioni esistenti tra gli algoritmi.

Da una più accurata analisi si è notato come non sia necessario tenere in considerazione tutte le coppie $\{user, item\}$ al fine del calcolo. Come viene mostrato in Figura 5.2, per la corretta esecuzione di PLIERS, è necessario considerare solo i vertici che rappresentano gli item con un cammino di lunghezza al più tre dall'utente target e i relativi utenti ad essi collegati. Indicando questi due insiemi di vertici rispettivamente con I_3 e U_{I_3}, con $|U_{I_3}| \leq n$ e $|I_3| \leq m$, la formula per il calcolo del vettore finale di PLIERS può essere riscritta come segue:

$$f_j = \begin{cases} \sum_{l \in U_{I_3}} \sum_{s \in I_3} \frac{1}{k(l) \cdot k(s) \cdot [k(j-s)+1]} & \text{se } j \in I_3 \\ 0 & \text{altrimenti} \end{cases}$$

Per mostrare l'efficacia dell'algoritmo proposto è stato appositamente realizzato un grafo bipartito user-tag composto da 64 utenti, 62 tag e un totale di 612 archi. Il grafo rappresenta gli interessi degli utenti nei confronti di vari argomenti, i quali sono caratterizzati da un grado di popolarità variabile. Si assume che un tag collegato a molti utenti si riferisca semanticamente a un argomento più generico rispetto a un tag poco popolare che si riferisce quindi a un argomento più specifico. Ad esempio, il tag *"TV-Series"* è molto più popolare rispetto a *"SherlockHolmes"* e, dal momento che la maggior parte degli utenti collegati al tag *"SherlockHolmes"* sono collegati anche a *"TV-Series"* (ma non sempre vale il contrario), semanticamente parlando, si intende il primo come "superclasse" del secondo. In Figura 5.3 viene mostrata la struttura del grafo bipartito, mettendo in risalto le caratteristiche di due casi limite: il primo (Utente 1), collegato solo a tag poco popolari (più specifici), e il secondo (Utente 3), collegato a tag la cui media di popolarità è più elevata. Di seguito vengono comparate le differenze nelle raccomandazioni effettuate da PLIERS, ProbS, HeatS e Hybrid per i due utenti considerati.

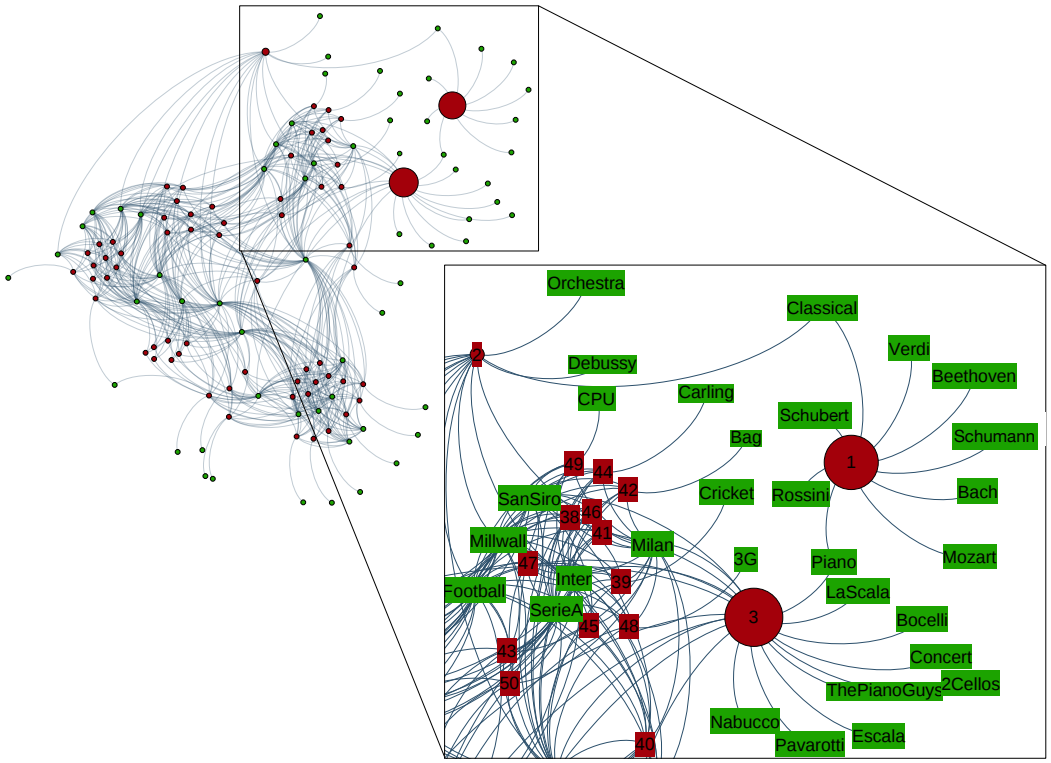


Figura 5.3: Struttura del grafo bipartito di esempio. I vertici di colore rosso rappresentano gli utenti e quelli verdi i tag.

Utente 1 L'Utente 1 è interessato ad argomenti legati alla musica classica (*"Schubert"*, *"Verdi"*, *"Rossini"*, etc...) che, nel grafo realizzato, hanno un grado di popolarità basso e la cui media corrisponde a 1.22. In Figura 5.4 viene mostrata la popolarità di ciascun tag collegato all'Utente 1.

La Figura 5.5 mostra invece i risultati ottenuti dall'esecuzione dei quattro algoritmi considerando l'Utente 1 come target. È stata riportata la classifica dei primi 10 tag consigliati da ciascun algoritmo e ordinati per popolarità. Le posizioni della classifica indicate vanno dalla numero 1 alla numero 25 per confrontare meglio le differenze tra i valori ottenuti.

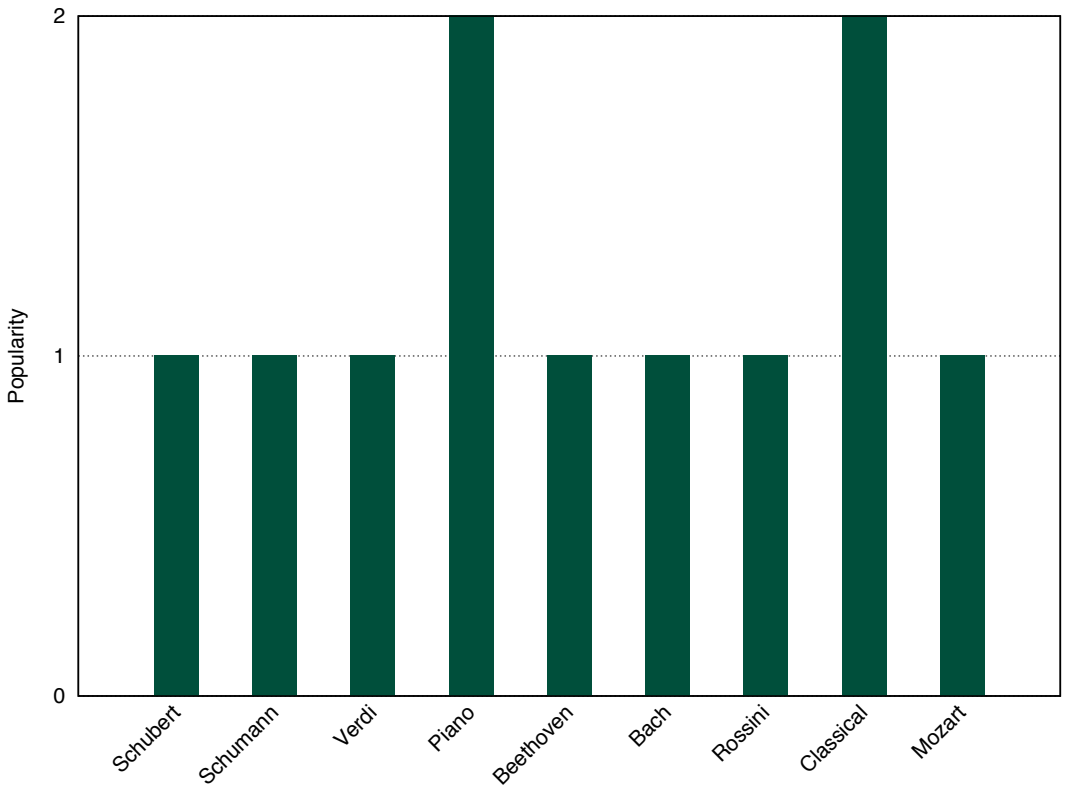


Figura 5.4: Popolarità dei tag direttamente collegati all'utente $U1$.

Si nota come in questo caso PLIERS, HeatS e Hybrid concordino sui primi 12 tag da suggerire all'utente, mettendo in risalto i tag con un grado di popolarità in linea con quelli dell'utente e per lo più semanticamente correlati (*"Debussy"*, *"Orchestra"*, *"Concert"*, etc...); dalla posizione 13, Hybrid inizia a divergere da PLIERS e Heats, assegnando un valore superiore agli item esaminati. ProbS, invece, sbaglia fin dalla prima posizione assegnando un punteggio maggiore a tag con una popolairtà che si discosta troppo da quelli in possesso dell'utente target e del tutto scorrelati dal punto di vista semantico (*"Football"*, *"StarTrek"*, *"Sport"*, etc...).

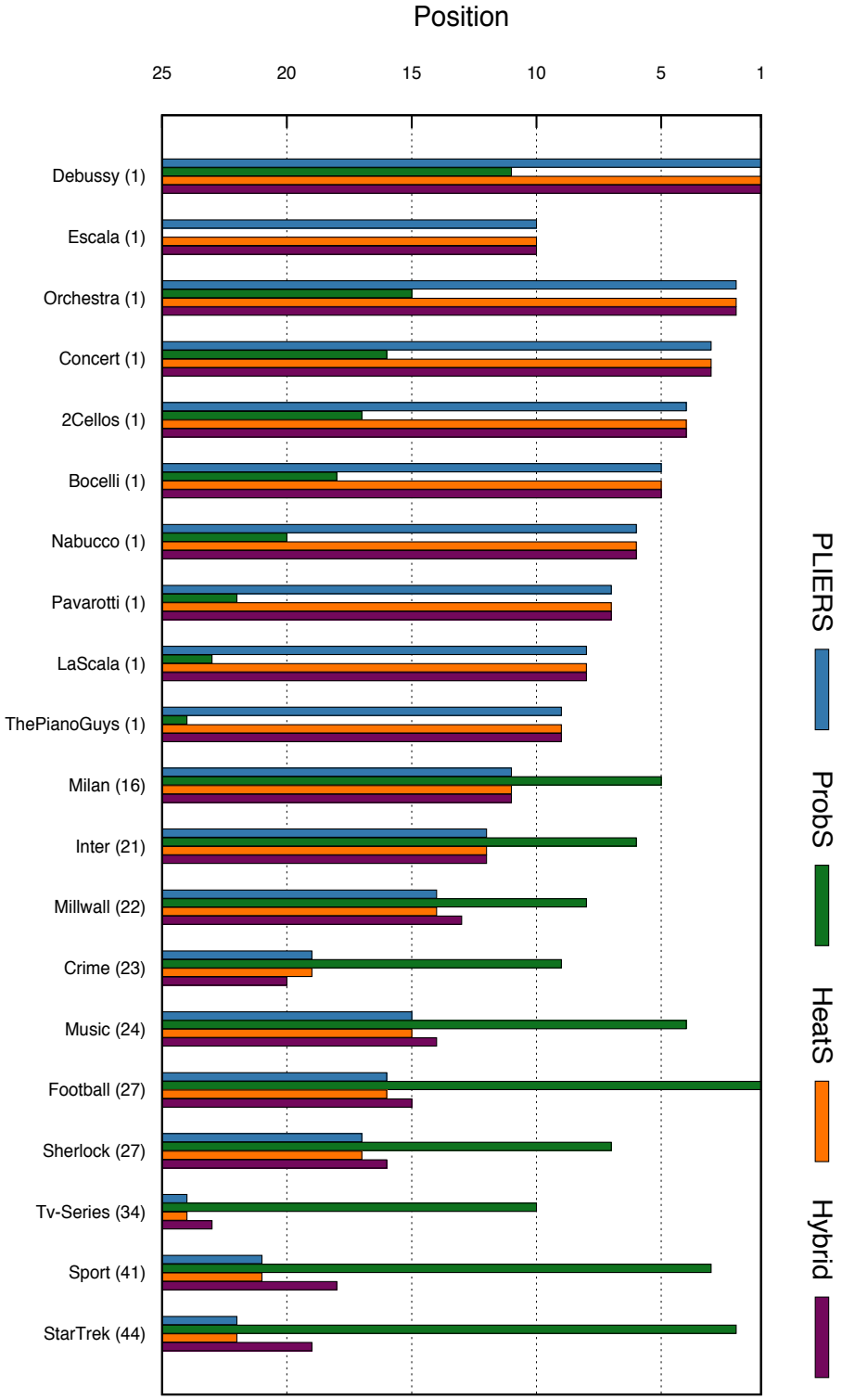


Figura 5.5: Tag consigliati all'utente $U1$.

Utente 3 L'Utente 3 è, invece, interessato sia a tag poco popolari (come, ad esempio,*"Pavarotti"*, *"2Cellos"*, *"Nabucco"*, etc...), ma anche a tag particolarmente popolari (*"StarTrek"*, *"StarWars"*, *"Millwall"*) e "generici" (*"Sport"*, *"Music"*) i quali alzano la media di popolarità degli argomenti a cui è interessato al valore di 15.3.

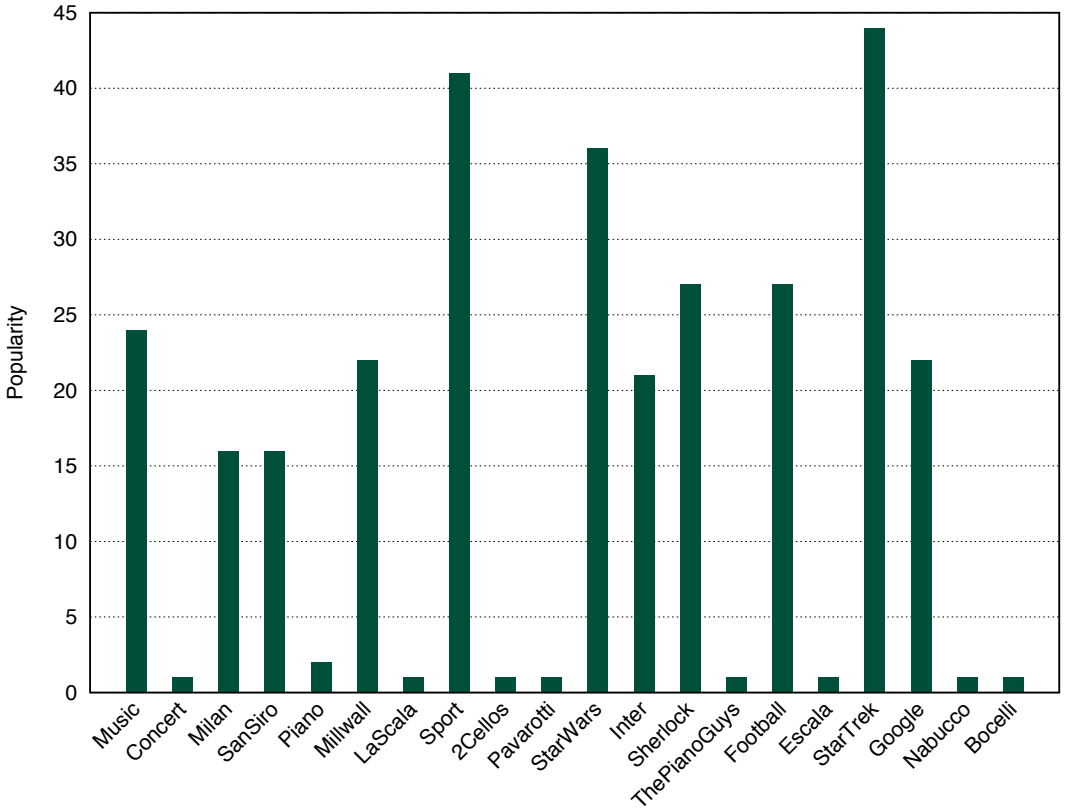


Figura 5.6: Popolarità dei tag direttamente collegati all'utente $U3$.

In Figura 5.7 sono riportati i risultati delle raccomandazioni ottenuti dai quattro algoritmi. In questo caso PLIERS si discosta sia dai risultati mostrati da HeatS, sia, in parte, da quelli ottenuti da ProbS e Hybrid, suggerendo quei tag la cui popolarità si discosta meno dalla media dell'utente. HeatS, mettendo sempre in risalto gli item poco popolari, consiglia principalente tag con popolarità pari a 1.

I risultati ottenuti da ProbS e Hybrid, invece, sono più simili tra loro, preferendo i tag più popolari ma, in alcuni casi, semanticamente non corretti secondo lo scopo prefissato. È interessante notare i risultati ottenuti per alcuni tag come *"Sci-Fi"*, *"TV-Series"* e *"Crime"*: seppur l'utente abbia mostrato interesse per note serie televisive (*"StarTrek"* e *"Sherlock"*), PLIERS, a differenza di ProbS e Hybrid, non suggerisce tag che si riferiscono ad argomenti correlati ma più generici, mettendo invece in risalto un altro tag, *"SherlockHolmes"*, semanticamente molto più in linea con gli interessi dell'utente.

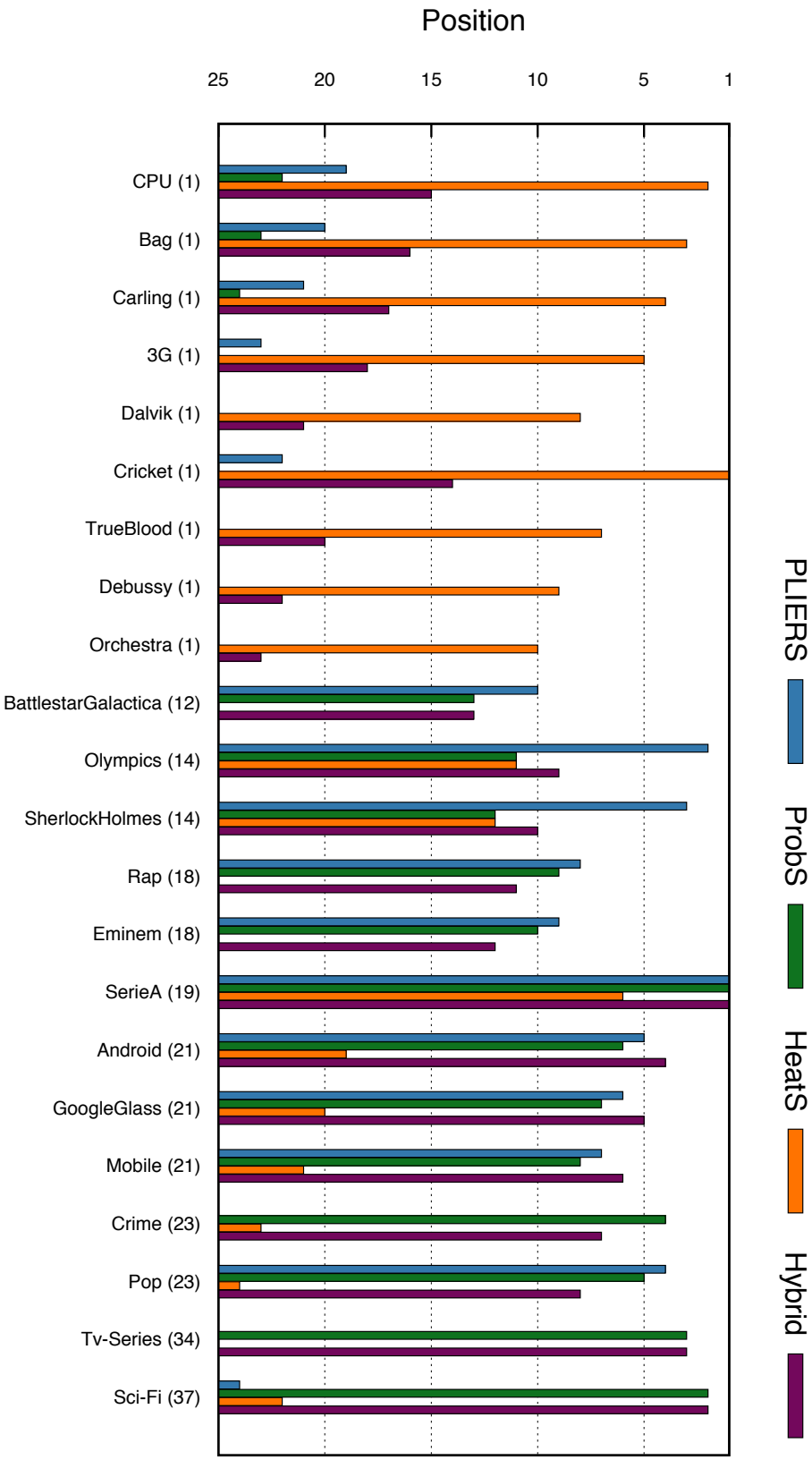


Figura 5.7: Tag consigliati all'utente $U3$.

5.2 PLIERS con un grafo tripartito

Uno degli scopi prefissati di questo lavoro, consiste nel fatto che l'algoritmo qui definito debba essere in grado di valutare l'utilità di un contenuto (o risorsa) per l'utente locale, sia in base ai tag a cui è stato associato, sia in base agli utenti che hanno espresso interesse verso tale item.

Analogamente a quanto effettuato in [62] con *ProbS*, anche *PLIERS* può essere applicato a un grafo tripartito *user-item-tag* (Figura 5.8). Ciascun arco $a_{i,j}$, tra l'utente U_i e l'item I_j, indica che l'item I_j è stato generato o consultato dall'utente U_i; l'arco $a'_{j,z}$, tra l'item I_j e il tag T_z, indica che l'item I_j è stato marcato con il tag T_z.

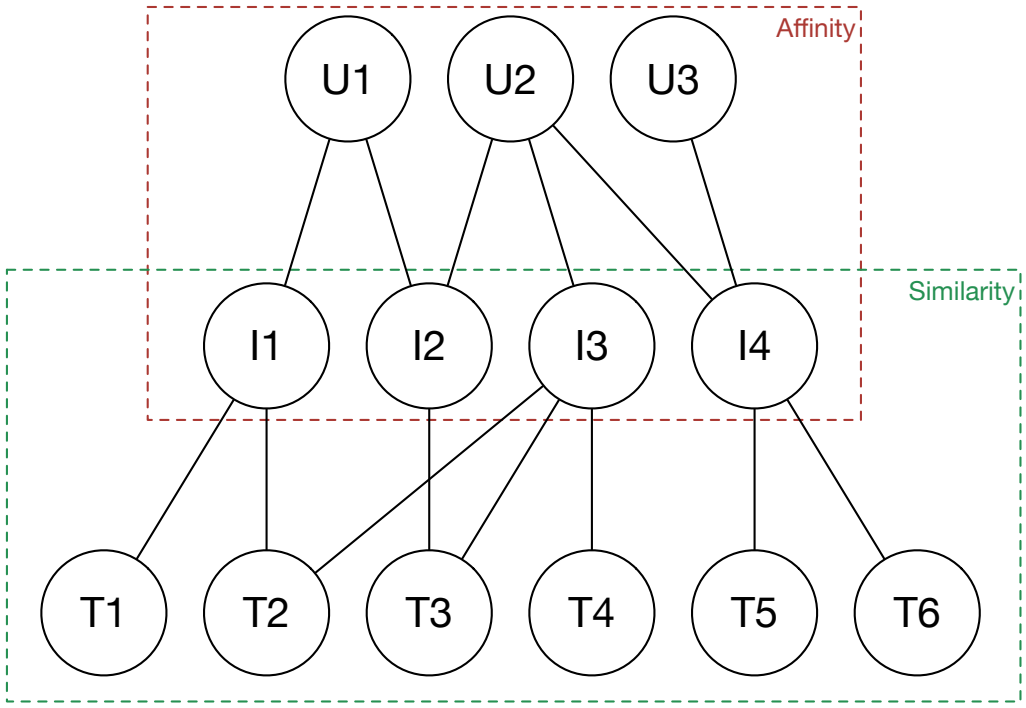


Figura 5.8: Indici di similarità e affinità.

A tal proposito, sono stati definiti due indici caratteristici:

Indice di affinità : calcolato tra i vertici user-item; indica quanto un dato contenuto risulta essere affine agli interessi dell'utente.

Indice di similarità calcolato tra i vertici item-tag; indica quanto due item siano simili in base ai tag con i quali sono stati marcati.

Per calcolare il grado di interesse di un utente nei confronti di uno specifico item I_j è possibile quindi effettuare la combinazione lineare degli indici di affinità e similarità, entrambi calcolati per I_j, nel seguente modo:

$$f_j^* = \lambda \cdot f_j^a + (1-\lambda) \cdot f_j^s \tag{5.2}$$

dove f_j^a e f_j^s sono rispettivamente l'indice di affinità e l'indice di similarità calcolati per l'item I_j e $\lambda \in [0,1]$ è un parametro dell'algoritmo da stimare.

5.3 Modalità opportunistica

Una volta implementato in CAMEO, PLIERS dovrà operare in un ambiente opportunistico. Il grafo tripartito user-item-tag, contenente le informazioni riguardanti gli interessi di tutti gli utenti e dei contenuti da essi condivisi, viene definito con il termine *grafo di conoscenza globale* (*global knowledge*). In una rete opportunistica ciascun nodo (coppia utente-dispositivo, indicato anche come *agente*) possiede solo una visione parziale del grafo globale (*grafo di conoscenza locale*, *local knowledge*), costruita in base alle informazioni locali e da quelle rilevate dai nodi incontrati.

Ad ogni contatto opportunistico tra due nodi, questi si scambieranno i reciproci grafi di conoscenza in modo tale che ciascuno di essi potrà aggiornare la propria local knowledge utilizzando le informazioni ricevute dall'altro. Terminata l'operazione di *update* del grafo locale, ciascun agente potrà valutare l'utilità dei nuovi item scoperti (se presenti) tramite PLIERS e, nel caso, richiederne il download.

In ambiente opportunistico, PLIERS non sarà utilizzato per stabilire una classifica di tutti gli item presenti nel grafo locale, ma verrà impiegato per calcolare l'utilità dei soli nuovi item scoperti. Tale calcolo avverrà come segue:

1. Ad ogni nuovo item verrà associata una risorsa (di valore 1) la quale verrà distribuita solo verso gli item collegati all'utente target ($U1$ in Figura 5.9).

2. Verrà calcolata la media degli indici di affinità e similarità tra il nuovo item considerato e gli item dell'utente target. Sempre in riferimento alla Figura 5.9:

   $AVG(f^a_{I3}) = \frac{f^a_{I3,I2}+f^a_{I3,I1}}{2} = \frac{\frac{1}{3}+0}{2} = \frac{1}{6}$

   $AVG(f^s_{I3}) = \frac{f^s_{I3,I2}+f^s_{I3,I1}}{2} = \frac{\frac{1}{18}+\frac{1}{18}}{2} = \frac{1}{18}$

3. Infine, il valore finale dell'item è ottenuto tramite la combinazione lineare dei due indici (Equazione 5.2):

   $f^*_{I3} = \lambda \cdot AVG(f^a_{I3}) + (1-\lambda) \cdot AVG(f^s_{I3})$

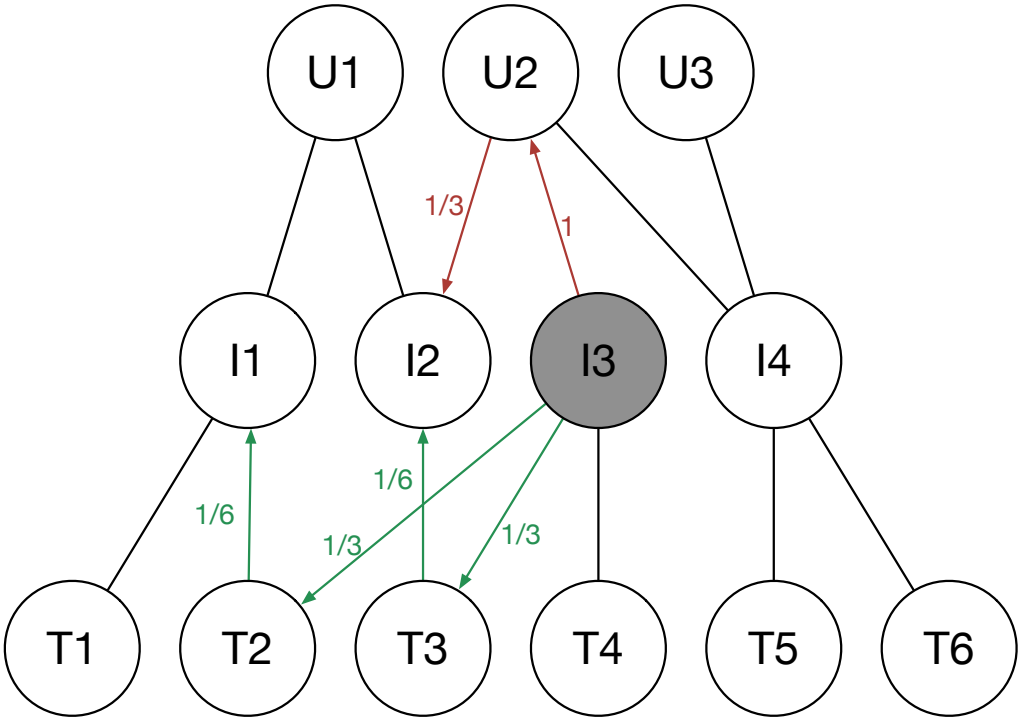


Figura 5.9: Calcolo degli indici di affinità (in rosso) e di similarità (in verde) per un singolo item ($I3$). In questo caso l'utente target è $U1$.

5.3.1 Buffer di suggerimento

Come è già stato sottolineato, gli item che ottengono un valore finale nullo non vengono ritenuti interessanti per l'utente locale dal momento che non sono diret-

tamente collegati nè ai suoi tag nè a utenti con interessi simili ai suoi. Per questo motivo, tali item non vengono presi in considerazione da PLIERS ai fini della raccomandazione.

Nell'ottica di permettere agli utenti di scoprire sempre nuovi contenuti, anche se ad un certo istante di tempo possano risultare di scarso interesse (ma non nullo) per l'utente locale, è necessario un meccanismo che sia adattivo in base ai contenuti scoperti dall'agente durante i propri contatti opportunistici.

A tale scopo viene proposta una soluzione basata sull'utilizzo di un *buffer di suggerimento*. Ciascun agente sarà dotato di un buffer di lunghezza prefissata in cui saranno posti tutti gli item (e relativi valori calcolati da PLIERS) che avranno ottenuto un risultato non nullo. Ogni volta che il buffer si riempie, viene calcolata la seguente formula:

$$current\_avg = (1-\beta) \cdot old\_avg + \beta \cdot new\_avg \tag{5.3}$$

dove old_avg si riferisce al valore di $current\_avg$ calcolato al riempimento precedente, new_avg è la media dei valori degli item al momento presenti nel buffer e β è un parametro dell'algoritmo con cui è possibile indicare quanto tenere conto degli item esaminati in passato rispetto a quelli correnti. In altre parole, ad ogni riempimento del buffer viene calcolata la media ponderata tra il valore medio degli item correnti e quelli esaminati al ciclo precedente. Il valore di $current\_avg$ così calcolato servirà per capire se consigliare o meno all'utente gli item che verranno scoperti con i futuri contatti. Infatti, se un futuro item verrà valutato da PLIERS con un valore maggiore o uguale a $current\_avg$, ne verrà consigliato il download all'utente e verrà aggiunto al buffer per il calcolo della prossima media. In questo modo l'algoritmo è in grado non solo di consigliare gli item che risultano più interessanti per l'utente locale, ma anche di adattarsi ai contenuti che vengono condivisi in rete.

In Figura 5.10 viene illustrato un esempio di funzionamento del meccanismo appena esposto, supponendo che sia stato appena stato svuotato il buffer e calcolato il nuovo valore di $current\_avg$.

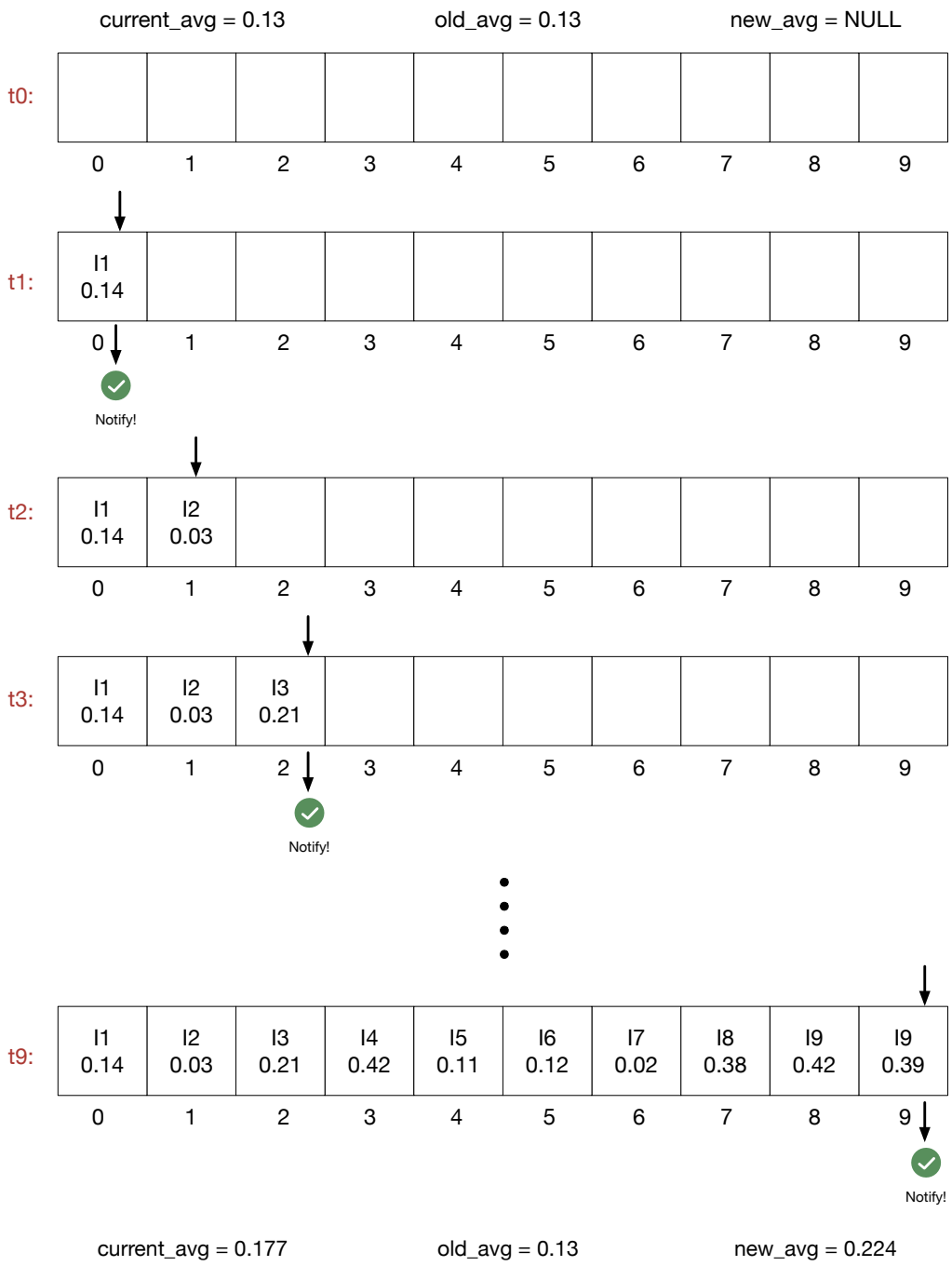


Figura 5.10: Esempio di funzionamento del buffer. Tutti gli item che riceveranno un valore superiore a *current_avg* (0.13) verranno suggeriti per il download. Per il calcolo del nuovo valore di *current_avg* è stato utilizzato $\beta = 0.5$.

5.3.2 Controllo del grafo di conoscenza locale

Con il crescere del numero di contatti opportunistici è lecito supporre che le informazioni contenute nel grafo locale continuino ad aumentare. Per evitare una crescita incontrollata del grafo, vengono proposte due possibili soluzioni:

Update limitato Durante la fase di update del grafo locale, cioè durante i contatti opportunistici, invece di aggiungere tutte le informazioni ricevute dall'altro nodo, è possibile includere solo gli item (e relativi user e tag ad essi collegati) il cui cammino minimo tra essi e l'utente target è di lunghezza inferiore o uguale a tre. Per quanto esposto nella Sezione 5.1, gli item con un cammino minimo dal target di lunghezza maggiore di 3 otterranno un valore nullo e, pertanto, considerati non interessanti per l'utente locale. In questo modo è possibile controllare meglio la crescita del grafo, evitando di occupare la memoria del dispositivo locale con informazioni che al momento non risultano essere di alcuna utilità per l'utente.

Un risvolto negativo di questo approccio consiste nel fatto che, scartando alcuni nodi, è probabile che vengano ignorate delle informazioni che potrebbero essere utili in futuro per l'utente locale.

I due approcci, che potrebbero essere identificati come *greedy* il primo (perchè fa la scelta migliore al tempo corrente) e *lungimirante* il secondo, potrebbero essere implementati entrambi e utilizzati in casi diversi: il primo quando il dispositivo ha poca memoria disponibile e il secondo in caso contrario.

Pruning del grafo Nel caso in cui il grafo locale raggiungesse una dimensione massima prefissata (intesa come somma del numero di vertici e archi), è necessario eliminare le informazioni che hanno meno probabilità di essere utili in futuro. Ad esempio, in [33] vengono utilizzati due metodi che, sfruttando il *principio di località temporale*, possono essere utili ad identificare le informazioni da mantenere nel grafo: *Most Recent Contacts (MR)* e *Most Frequent Contacts (MF)*. In MR si assume che le informazioni più vecchie abbiano meno potere predittivo rispetto a quelle aggiunte più di recente. In questo caso quindi, i vertici e gli archi del grafo devono essere associati al timestamp corrispondente al momento in cui

vengono aggiunti (o ri-aggiunti) al grafo locale. In questo caso, quindi, l'operazione di *prune* del grafo consiste nell'eliminare tutti i vertici e gli archi con un timestamp più vecchio rispetto a un t_{max} definito a priori. Nel caso di MF vengono invece mantenute nel grafo le sole informazioni utilizzate più frequentemente dall'algoritmo. A tale scopo i vertici e gli archi dovranno essere quindi associati a un contatore che tiene traccia del numero di volte che vengono utilizzati da PLIERS per effettuare le raccomandazioni.

Indipendentemente dal metodo di *pruning* scelto, è importante che le informazioni rimanenti nel grafo locale mantengano un certo livello di consistenza. In altre parole, ai fini della corretta esecuzione di PLIERS, è necessario che alla fine dell'operazione di pruning non rimangano vertici sconnessi da altre componenti del grafo. Facendo riferimento all'esempio riportato in Figura 5.11: se, per qualche politica, dovesse essere eliminato il vertice $U2$ (Utente 2), dovranno essere eliminati anche gli item e i relativi tag ad esso collegati e che non risultano utili a nessun altro utente.

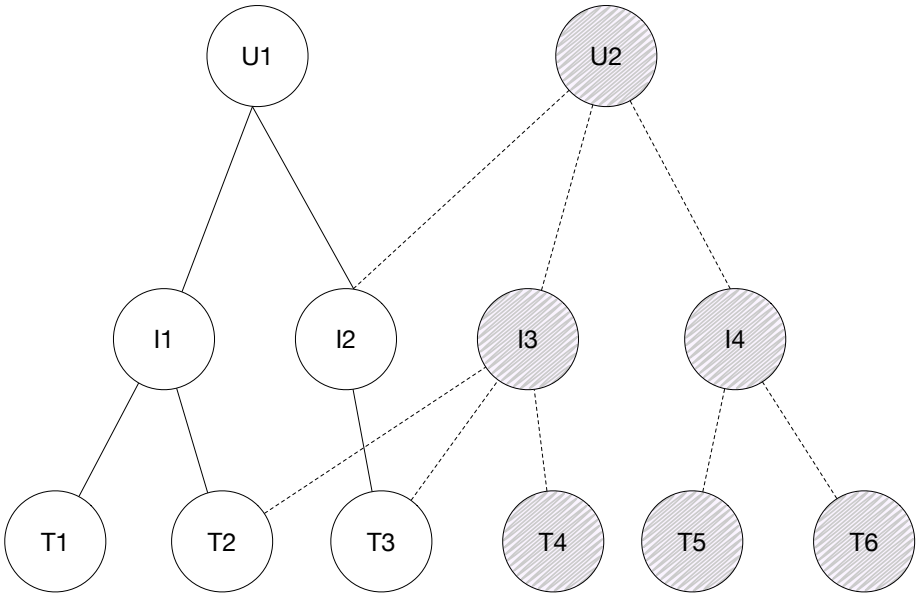


Figura 5.11: *Prune* del grafo locale. I vertici e gli archi che verranno eliminati considerando $U2$ come l'utente incontrato meno di recente sono evidenziati con linee tratteggiate.

Capitolo 6

Simulazioni e risultati

In questo capitolo viene inizialmente introdotto l'insieme dei dati (dataset) utilizzati negli esperimenti e il processo utilizzato per campionare gli stessi. Successivamente, vengono presentati i risultati dei test eseguiti al fine di verificare l'effettiva validità di PLIERS per gli scopi prefissati. Tali test possono essere divisi in due categorie:

Ranking I test hanno lo scopo di dimostrare l'efficacia di PLIERS nel suggerire item potenzialmente interessanti per l'utente target. In questo caso i dati vengono rappresentati con un grafo bipartito user-tag; l'algoritmo genera una classifica dei tag in ordine di rilevanza per l'utente che, successivamente, viene confrontata con le classifiche calcolate dagli altri algoritmi considerati: ProbS, HeatS e Hybrid.

Simulazioni opportunistiche Vengono simulati dei contatti opportunistici tra un numero prefissato di nodi (qui indicati anche con il termine *agenti*). In queste simulazioni viene analizzato quanto il grafo di conoscenza costruito da ciascun nodo differisca da quello globale, costruito sulla base dei campioni estratti dal dataset a disposizione. Simulando la generazione di contenuti nella rete, viene inoltre misurata la precisione con cui i contenuti vengono selezionati dagli agenti rispetto alla visione globale del grafo di conoscenza.

In questo caso, sia il grafo di conoscenza globale, sia quello locale, vengono rappresentati utilizzando dei grafi tripartiti user-item-tag.

Il middleware CAMEO è stato sviluppato per essere eseguito su dispositivi Android[1]; per questo motivo PLIERS e le strutture dati utilizzate (grafo bipartito e tripartito) sono state implementate nel linguaggio di programmazione Java.

La macchina e il dataset utilizzati per effettuare i test riportati in questo capitolo sono stati messi a disposizione dall'Istituto di Informatica e Telematica (IIT) del Consiglio Nazionale delle Ricerche (CNR) di Pisa. Le specifiche della macchina sono riportate in Tabella 6.1.

Tabella 6.1: Specifiche della macchina usata per i test.

CPU	64 x 2.6 GHz
RAM	130 GB
Filesystem	16 TB
Architettura	x86_64
S.O.	Linux 3.8.13 (Gentoo)
Java	1.7.0_45

6.1 Il dataset

I dati utilizzati per effettuare le valutazioni sono stati raccolti dall'IIT del CNR di Pisa allo scopo di svolgere degi studi relativi alle reti sociali (online social network).

Il dataset è costituito dalle informazioni prelevate dal social network Twitter[2] inerenti le attività (tweet e re-tweet) di 1630594 utenti relative a un arco temporale di circa due anni (dal 2012 al 2014).

Tabella 6.2: Statistiche del dataset Twitter.

Users	**Items**	**Tags**
1630594	405092911	30156233

[1] https://www.android.com
[2] https://twitter.com

Nell'Esempio 6.1 vengono riportate le informazioni presenti nel dataset relative all'utente con identificativo 12. Le informazioni inerenti ai tweet sono separate dal carattere *"#"* e si riferiscono rispettivamente a: identificativo, timestamp di creazione (in formato UNIX) e i tag associati. Ad esempio, alle ore *17:14* del giorno *10/17/2012*, l'utente 12 ha creato un tweet con il tag *12in12* a cui è stato assegnato l'identificativo *25861670718946*.

UserId	Tweets
12	25861670718946#1350494075#12in12
	25801651002136#1350350904#Postseason
	25737232986736#1350197526#grateful

Esempio 6.1: Tweet generati dall'utente con id 12

In Figura 6.1 viene illustrata, in scala logaritmica, la CCDF (*Complementary Cumulative Distribution Function*) della distribuzione delle frequenze di utilizzo (popolarità) dei tag presenti nel dataset. La funzione CCDF indica la probabilità, $P(X > x)$, che una certa variabile casuale possa assumere un valore superiore a un dato x. In altre parole, in questo caso, indica la probabilità che la frequenza di utilizzo dei tag presenti nel dataset possa essere superiore a un certo valore x. Dal grafico di Figura 6.1 si può quindi dedurre che, all'interno del dataset considerato, i tag con popolarità bassa sono numericamente superiori rispetto a quelli caratterizzati da una frequenza di utilizzo più elevata. Questa distribuzione è simile a quella utilizzata in letteratura per descrivere diverse caratteristiche delle reti sociali e viene indicata con il nome di *power law* [19, 4].

In Figura 6.2 sono state riportate le frequenze di utilizzo dei primi 30 tag più popolari presenti nel dataset di riferimento.

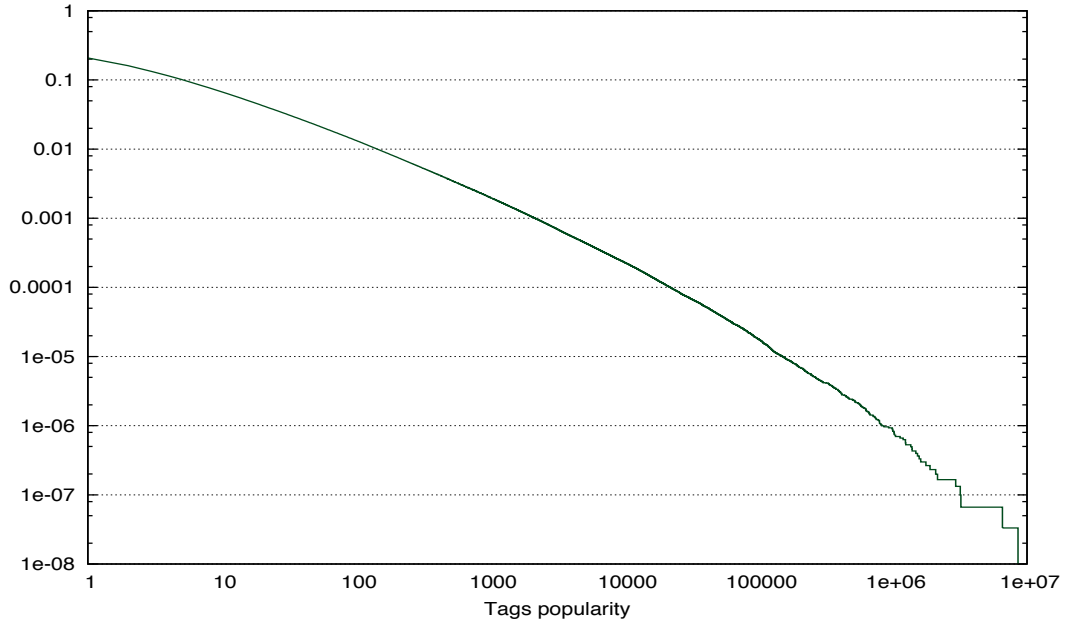


Figura 6.1: CCDF, in scala logaritmica, delle frequenze di utilizzo dei tag.

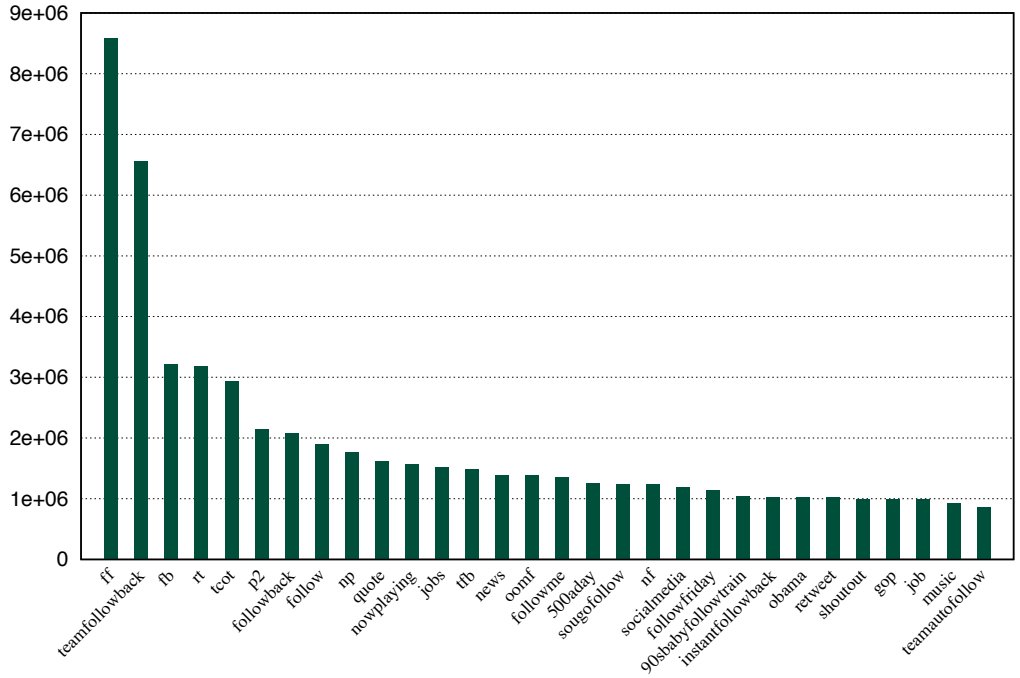


Figura 6.2: Popolarità dei primi 30 tag nel dataset Twitter.

6.2 Campionamento dei dati

Seppur la macchina utilizzata per eseguire i test sia dotata di una gran quantità di risorse computazionali, non è risultato praticabile mantenere in memoria RAM l'intero dataset. Per questo motivo, in tutti i test e simulazioni opportunistiche eseguite, è stato necessario campionare i dati in modo tale da elaborare solo un sotto-insieme delle informazioni contenute nel dataset.

Dal momento che nel dataset utilizzato possono essere presenti degli utenti che non hanno item condivisi con altri utenti (re-tweet), al fine di limitare il più possibile l'estrazione di "componenti sconnesse", per il campionamento dei dati è stato deciso di utilizzare la tecnica nota in letteratura con il nome di *Snowball sampling* [9]. Tale tecnica può essere espressa nel seguente modo: per estrarre un campione di K utenti viene, innanzitutto, selezionato un utente in maniera casuale (u_{random}) e aggiunto (insieme ai suoi item e tag) al campione; successivamente vengono selezionati tutti gli utenti che hanno degli item in comune con u_{random} e vengono aggiunti anch'essi (e relativi item e tag) al campione. Il procedimento viene reiterato fino a raggiungere la taglia K dell'insieme desiderato. Nel caso in cui un utente non abbia nessun item in comune con altri, viene ri-estratto un utente in maniera casuale.

In Figura 6.3 viene riportato un semplice esempio che mostra l'estrazione di un insieme di $K = 3$ utenti tramite la tecnica Snowball. In verde vengono indicate le informazioni estratte durante la $i-esima$ iterazione dell'algoritmo e in blu gli utenti candidati all'estrazione e messi in una coda di priorità. Durante la prima iterazione viene selezionato, in maniera randomica, l'utente $U5$ e posto, insieme ai suoi item e relativi tag, all'interno del sotto-insieme estratto. Gli utenti $U6$ e $U2$ vengono posti nella coda di priorità. Alla seconda iterazione, viene estratto il primo utente presente nella coda, $U6$, e viene aggiunto a quest'ultima l'utente $U4$ dal momento che risulta collegato a un item ($I7$) posseduto da $U6$. Infine, all'ultima iterazione, vengono estratti i dati relativi all'utente $U2$ e posti nella coda di priorità gli utenti $U1$ e $U3$. A questo punto la procedura può terminare dal momento che il campione ha raggiunto la dimensione desiderata.

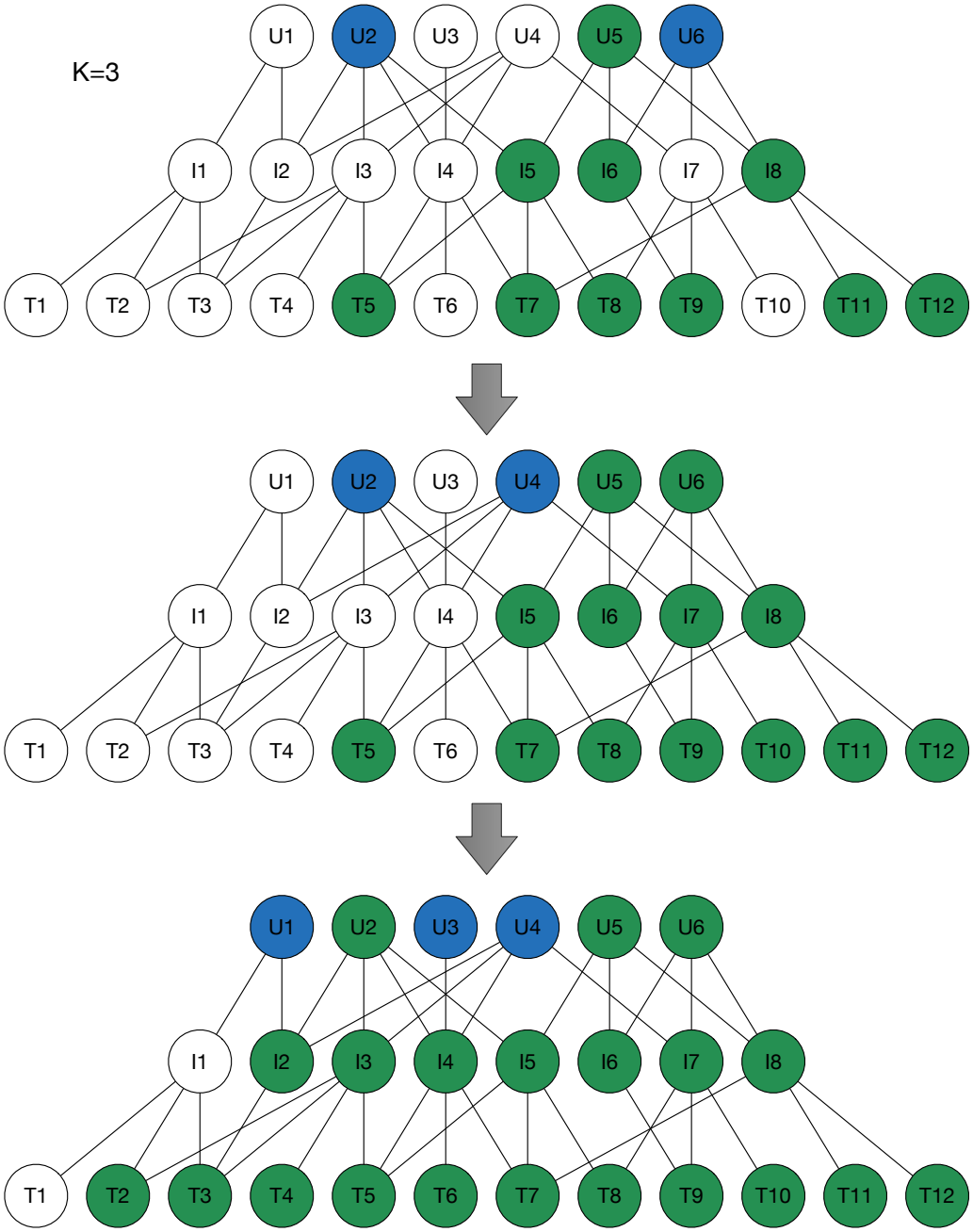


Figura 6.3: Esempio di campionamento con la tecnica *Snowball*. In verde vengono indicate le informazioni già estratte e in blu gli utenti che sono in lista per essere estratti.

6.3 Complessità in tempo

Per testare i tempi di esecuzione di PLIERS in "modalità opportunistica", cioè esaminando un numero variabile di nuovi item "scoperti" durante un contatto tra due nodi della rete, è stato utilizzato un grafo tripartito user-item-tag costruito sulla base di un campione di 5000 utenti. Le caratteristiche del grafo globale sono riportate in Tabella 6.3. L'utente target è stato estratto casualmente dall'insieme di tutti gli utenti presenti nel grafo. Indicando con T l'insieme di tutti i 713244 item, per simulare la "scoperta" di nuovi contenuti, sono stati proposti a PLIERS dei sottoinsiemi di item, $t \subseteq T$, di cardinalità crescente. In questo modo è stato possibile rilevare i tempi di esecuzione dell'algoritmo al variare del numero di item sottoposti.

Tabella 6.3: Statistiche del grafo tripartito utilizzato per testare la complessità in tempo di PLIERS in ambiente opportunistico.

Users	**Items**	**Tags**	**Edges**
5000	713244	190865	1745119

In Figura 6.4 vengono riportati i tempi di esecuzione in millisecondi (*ms*) per un numero variabile di item tra 0 e 45000. Come è possibile notare, la complessità in tempo dell'algoritmo risulta essere lineare nel numero di item esaminati e il tempo massimo ottenuto è poco superiore ai 16 secondi.

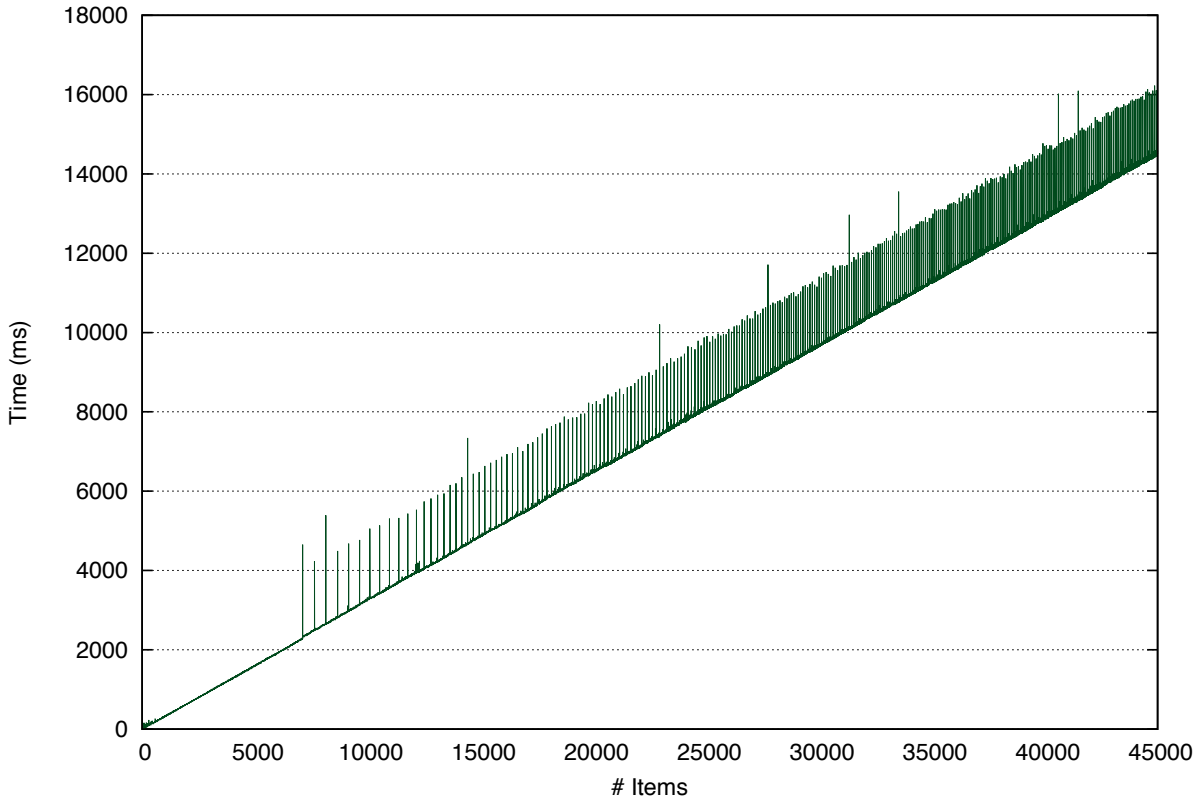


Figura 6.4: Tempi di esecuzione di PLIERS al variare del numero di item da analizzare.

6.4 Ranking con grafo bipartito

Nel Capitolo 5.1 è stato proposto un esempio per mostrare l'efficacia di PLIERS nel suggerire tag che fossero semanticamente correlati a quelli già in possesso dell'utente target. Al fine di ottenere dei risultati più consistenti, è stato effettuato un esperimento analogo utilizzando un campione di 5000 utenti per creare il grafo bipartito user-tag.

Tabella 6.4: Statistiche del grafo bipartito user-tag.

Users	**Tags**	**Edges**
5000	194330	508465

In mancanza di parametri più precisi per valutare le differenze tra PLIERS, ProbS, HeatS e Hybrid, è stato analizzato, per ogni utente, come varia la popolarità dei primi 10 tag proposti da ciascun algoritmo rispetto alla popolarità dei tag già collegati all'utente target. In Figura 6.6 è raffigurato l'andamento della CCDF (in scala logaritmica) dello scarto quadratico medio (σ) tra la mediana di popolarità dei tag proposti e la mediana dei tag già in possesso degli utenti. Dal grafico è evidente come ProbS e HeatS propongano dei tag la cui popolarità si discosta meno da quelli dell'utente target rispetto ai contenuti consigliati dagli altri due algoritmi.

Un secondo parametro di confronto utilizzato riguarda il valore di *overlap* medio presente tra i tag proposti e quelli già in possesso dell'utente target. Per overlap tra due tag, t_1 e t_2, si intende il valore di similarità tra l'insieme degli utenti collegati a t_1 (U_{t_1}) e quello degli utenti collegati a t_2 (U_{t_2}). La similarità tra due insiemi viene calcolata utilizzando l'*indice di similarità di Jaccard*:

$$sim(U_{t_1}, U_{t_2}) = \frac{|U_{t_1} \cap U_{t_2}|}{|U_{t_1} \cup U_{t_2}|} \tag{6.1}$$

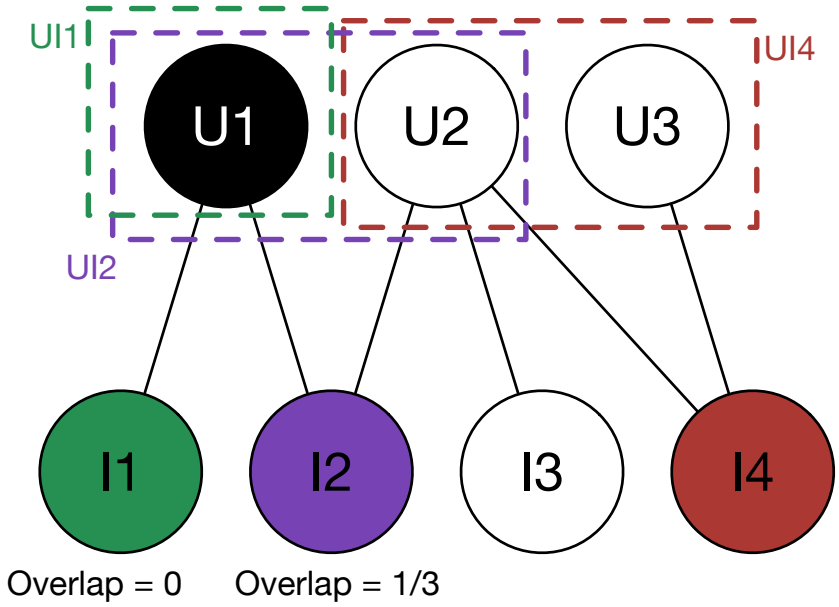


Figura 6.5: Esempio di calcolo del valore di Overlap per l'item $I4$.

In Figura 6.5 è rappresentato un esempio di calcolo dell'overlap medio tra l'item $I4$ e quelli dell'utente target $U1$. Innanzitutto viene calcolato l'indice di Jaccard (Formula 6.1) tra le due coppie di item $\{I4, I1\}$ e $\{I4, I2\}$, rispettivamente $sim(U_{I_4}, U_{I_1} = 0)$ e $sim(U_{I_4}, U_{I_2} = \frac{1}{3})$; infine, il valore di Overlap per l'item $I4$ risulta essere:

$$Overlap(I4) = \frac{sim(U_{I_4}, U_{I_1}) \cdot sim(U_{I_4}, U_{I_2})}{2} = \frac{1}{6}$$

Seppur dal grafico dello scarto di popolarità possa sembrare che PLIERS e HeatS mostrino lo stesso comportamento, analizzando l'overlap medio per tutti gli utenti ci si accorge che in effetti non è così. Il grafico in Figura 6.7 dimostra, infatti, come PLIERS proponga dei tag caratterizzati da un valore di overlap medio nettamente superiore a quello dei tag consigliati da HeatS.

Per comparare gli algoritmi risulta, quindi, necessario considerare insieme entrambi gli indici proposti: differenze di popolarità e overlap medio. Infatti, osservando, ad esempio, il grafico di Figura 6.7, ProbS mostra un grado di overlap superiore rispetto agli altri algoritmi, ma questo è dovuto al fatto che i contenuti da esso suggeriti sono così popolari (e generici) che si sovrappongono a molti tag presenti nel dataset.

Per questo motivo, quindi, è possibile dire che, per il campione dei dati analizzato, PLIERS mostra un comportamento migliore rispetto gli altri algoritmi considerati per lo scopo prefissato: suggerire degli item interessanti per un dato utente basandosi sulla sua storia passata (interessi mostrati in passato).

Tabella 6.5: Statistiche dei risultati ottenuti dai quattro algoritmi.

	Popularity (AVG)	**Popularity (Median)**	**Overlap (AVG)**	**Overlap (Median)**
PLIERS	72.9676	18.5816	0.0108724	0.0114297
ProbS	557.434	621.921	0.0208316	0.0219806
HeatS	73.2231	18.6822	0.00127495	0.00146471
Hybrid	209.749	217.112	0.0196479	0.0213437

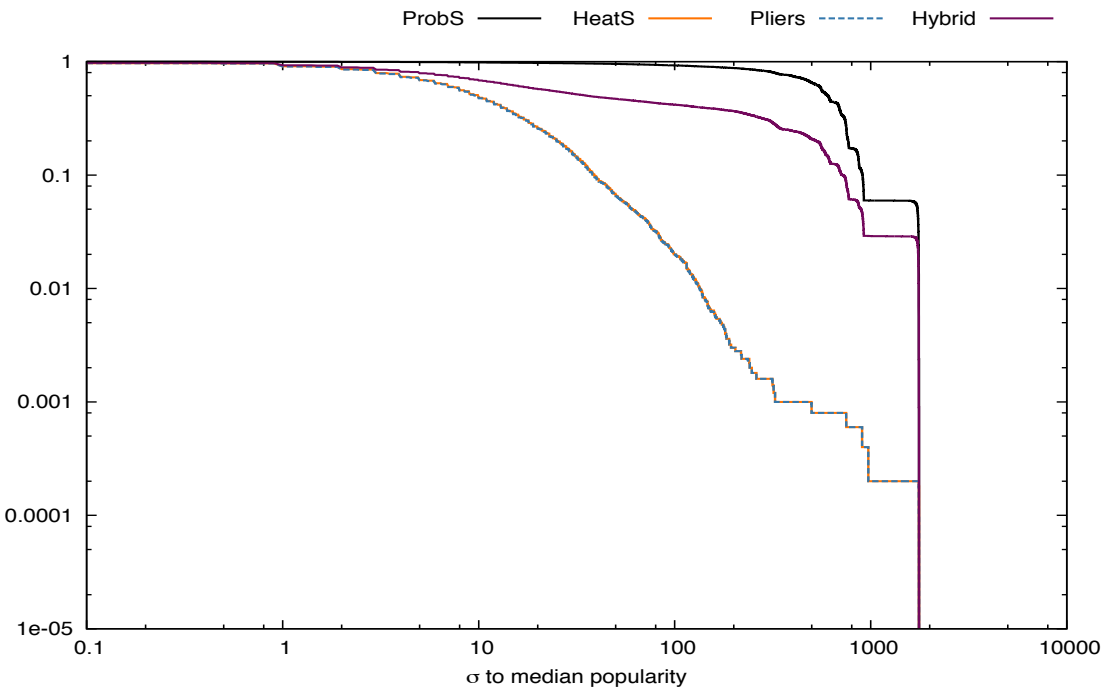


Figura 6.6: CCDF, in scala logaritmica, dello scarto tra la mediana di popolarità dei primi 10 tag proposti e la mediana dei tag già in possesso degli utenti.

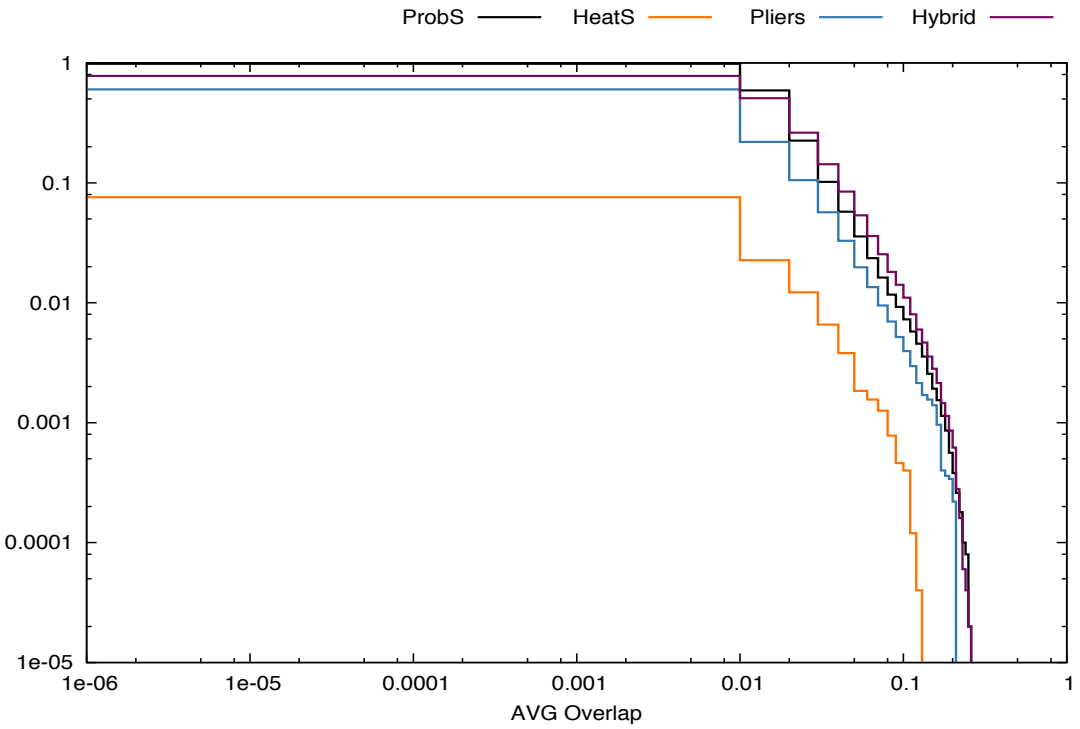


Figura 6.7: CCDF, in scala logaritmica, della mediana dell'overlap dei primi 10 tag proposti.

6.5 Simulazioni in ambiente opportunistico

Allo scopo di studiare il comportamento di PLIERS in ambiente opportunistico è stato realizzato un software il quale, dati in input delle tracce di mobilità e le informazioni relative a un numero fissato di agenti, simula le azioni intraprese da ciascuna coppia di nodi nel momento in cui avviene un contatto opportunistico.

In Tabella 6.6 sono riportate le caratteristiche dei campioni estratti dal dataset Twitter utilizzati nelle simulazioni opportunistiche.

Tabella 6.6: Statistiche campioni usati nelle simulazioni.

Users	**Items**	**Tags**
50	3193	2059
100	4520	3235
200	13213	7561

6.5.1 Il simulatore

Il simulatore è stato realizzato in maniera modulare in modo tale da risultare estendibile per testare futuri algoritmi di reasoning in ambiente opportunistico.

In Figura 6.8 viene riportato lo schema del simulatore realizzato, mettendo in evidenza i principali moduli di cui è costituito. Il package denominato *Top Layer* contiene i moduli che si occupano di leggere e processare i dati in input, impostare i parametri delle simulazioni e avviare queste ultime. I moduli contenuti in *Experiments* implementano le azioni che ciascun agente dovrà eseguire durante i contatti opportunistici che lo coinvolgono. In *Reasoners* sono presenti gli algoritmi di context-reasoning che possono essere utilizzati dagli agenti. Nel package *Opportunistic* risiedono i moduli che rappresentano le entità protagoniste dei contatti opportunistici: i contatti (*Contact*[3]) e gli agenti (*Agent*). Inoltre, ciascun agente, mantiene la propria conoscenza locale sotto forma di grafo (bipartito

[3]Per gli esperimenti eseguiti in questo lavoro di tesi sono state utilizzate le tracce di mobilità generate da un simulatore che implementa il modello di mobilità HCMM[15]

o tripartito). I moduli *Logger* e *File Manager* si occupano, rispettivamente, di tenere traccia delle azioni che avvengono durante le simulazioni e di interfacciarsi al filesystem per salvare, ad esempio, le statistiche riguardanti i dati scambiati tra gli agenti.

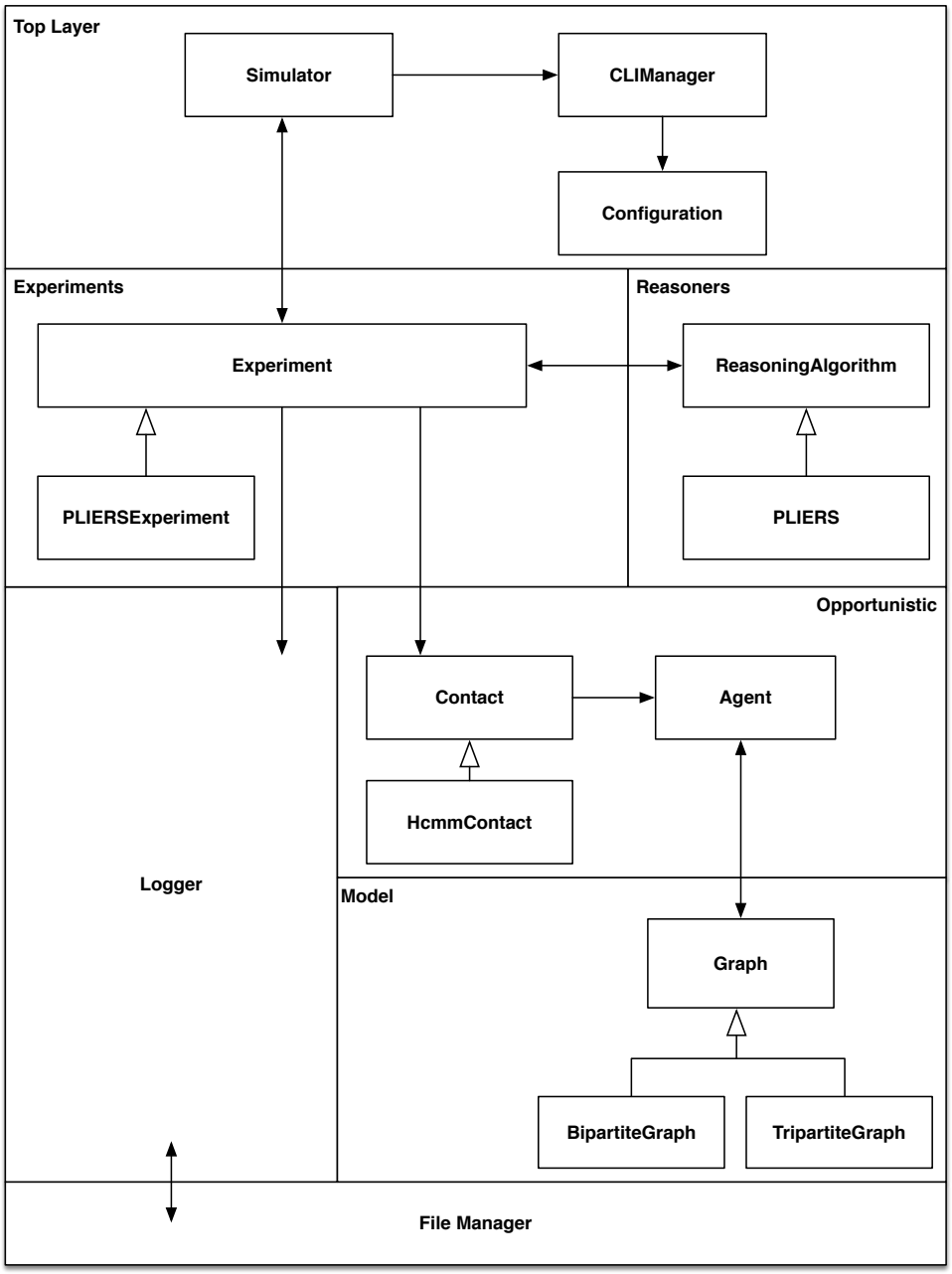


Figura 6.8: Schema del simulatore opportunistico.

Qui di seguito viene descritto il funzionamento del modulo che si occupa di simulare l'esecuzione di PLIERS in ambiente opportunistico: PLIERSExperiment. In questo tipo di simulazione non sono stati tenuti in considerazione vincoli come, ad esempio: l'effettiva durata dei contatti, limitata disponibilità di banda per trasmettere i dati e una dimensione finita della memoria di ciascun agente.

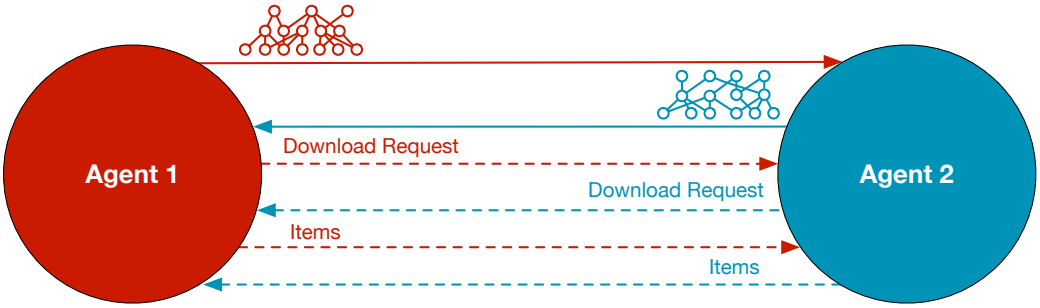


Figura 6.9: Azioni eseguite durante un contatto opportunistico.

Facendo riferimento alla Figura 6.9 e supponendo la simulazione di un contatto tra due generici agenti, *Agent 1* e *Agent 2*, le azioni compiute da entrambi sono le seguenti:

Scambio delle informazioni di contesto Ciascun agente invia all'altro il proprio grafo di conoscenza locale contenente le informazioni relative alla generazione/-download di item da parte sua e da parte dei nodi incontrati precedentemente.

Update conoscenza locale Ogni agente aggiorna il proprio grafo locale con le informazioni contenute in quello dell'altro nodo.

Identificazione nuovi item Ciascun agente mantiene una lista di item esaminati in passato. Analizzando le nuove informazioni ricevute dall'altro nodo, ciascuno di essi è in grado di identificare gli item che non sono ancora stati esaminati.

Esecuzione di PLIERS A questo punto viene eseguito PLIERS per valutare "quanto" i nuovi item possano essere interessanti per l'agente locale.

Download Infine viene richiesto il download di quegli item (se presenti) a cui è stato associato un valore di interesse non nullo da parte di PLIERS. Tali item ver-

ranno aggiunti nel buffer di suggerimento mantenuto da ciascun nodo e suggeriti secondo la politica esposta nel Capitolo 5.3.1.

6.5.2 Simulazioni senza download

In questa sezione vengono riportati i risultati relativi a un insieme di simulazioni effettuate allo scopo di studiare la crescita, all'aumentare del numero di contatti, del grafo locale di ciascun nodo rispetto al grafo di conoscenza globale. In questo caso, quindi, era necessario solo che gli agenti, durante i contatti opportunistici, si scambiassero le relative informazioni di contesto e aggiornassero i propri grafi di conoscenza locale.

Dal grafico rappresentato in Figura 6.10 è possibile notare come, dopo un numero opportuno di contatti, tutti gli agenti possiedano, in pratica, l'intero grafo di conoscenza globale. Questo risultato suggerisce che, probabilmente, ai fini della distribuzione dei contenuti in rete, non sia necessario che ciascun nodo mantenga l'intera storia passata dei nodi incontrati. Sarebbe interessante, quindi, studiare in futuro come potrebbe variare la distribuzione delle informazioni, controllando la crescita del grafo locale tramite le tecniche proposte nel Capitolo 5.3.2.

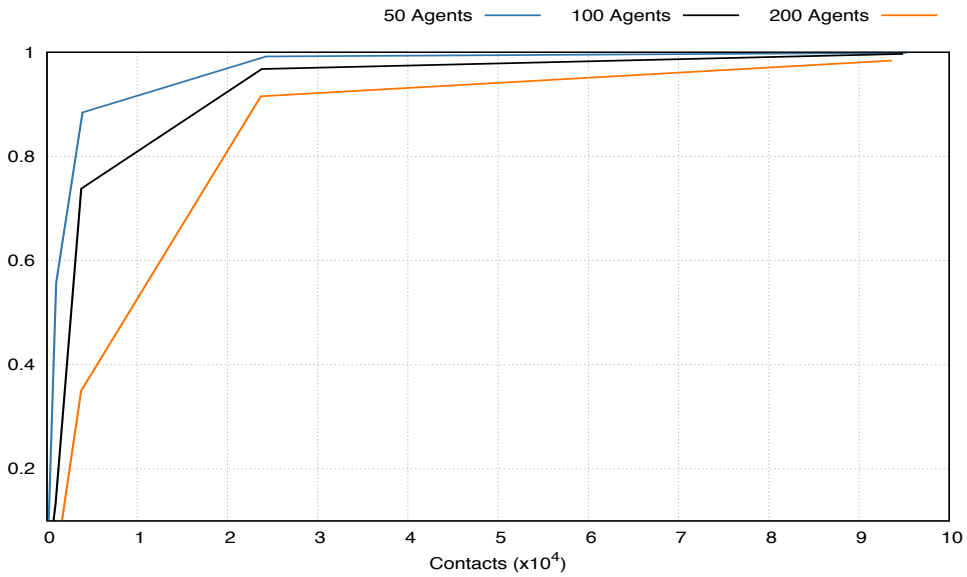


Figura 6.10: Similarità media del grafo locale rispetto alla conoscenza globale, al variare del numero di contatti e del numero di agenti in rete.

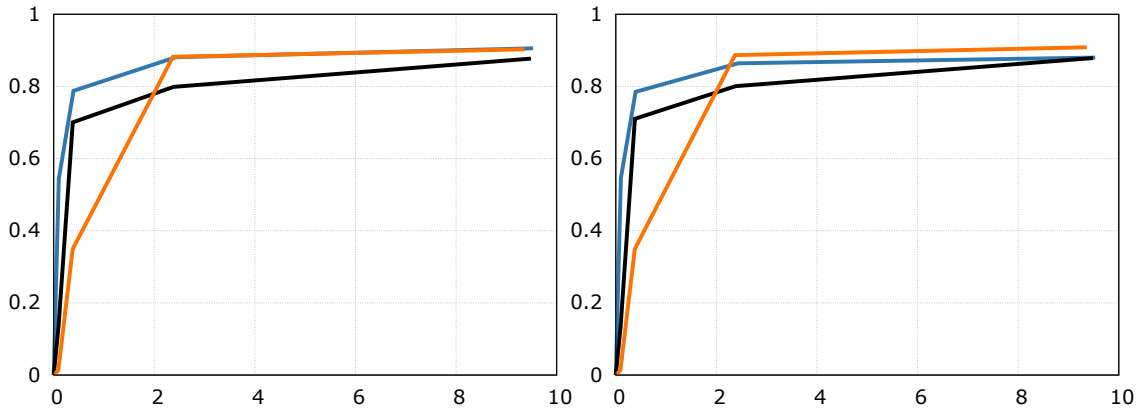
6.5.3 Simulazioni con download

I risultati riportati in questa sezione si riferiscono ad una serie di simulazioni in cui è stato permesso agli agenti di scaricare i contenuti scoperti durante i contatti opportunistici. Tali contenuti, che rappresentano i tweet del dataset descritto nella Sezione 6.1, sono associati a un timestamp di creazione. Rapportando tali timestamp alla finestra temporale di ogni singolo esperimento, è stato possibile simulare la generazione dinamica dei contenuti da parte degli agenti.

A differenza di quanto accadeva nelle simulazioni senza download, in questo caso è possibile notare (Figura 6.11) come il grafo locale degli agenti tenda più lentamente e, in alcuni casi, diverga dal grafo di conoscenza globale. Questo è dovuto al fatto che, permettendo agli agenti di scaricare i contenuti proposti da PLIERS durante le simulazioni, vengono aggiunti al grafo locale dei link (tra utenti e item) che non sono presenti nello scenario statico rappresentato dal grafo globale. Inoltre, nella stessa figura, è possibile notare come la grandezza del buffer di suggerimento (Capitolo 5.3.1), incida notevolmente sulla quantità di item scaricati e, di conseguenza, sull'andamento del grafo locale degli agenti.

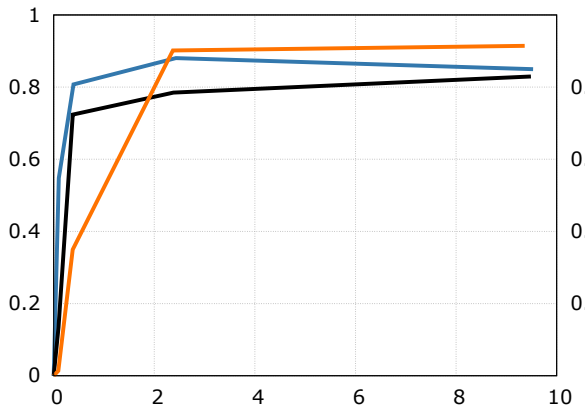
Sono state effettuate, inoltre, delle analisi riguardanti la tipologia degli item scaricati durante le simulazioni: per ogni agente è stata confrontata la lista degli item potenzialmente interessanti per esso (cioè, con un valore calcolato da PLIERS diverso da 0) nello scenario statico, definito come *resource vector globale*, con la lista degli item scaricati durante le simulazioni. In Figura 6.12 sono riportati i risultati ottenuti da tale confronto. Come è possibile notare, i download effettuati rappresentano solo poco più del 5% del resource vector globale. Il motivo di questo comportamento è da ricondursi, anche in questo caso, alla creazione di nuovi link nel grafo locale (dovuti ai download) e al meccanismo del buffer di suggerimento: l'azione di scaricare nuovi contenuti, creando così dei link che nel grafo globale non sono presenti, permette a PLIERS di assegnare un valore diverso da 0 a contenuti che, nello scenario statico, venivano invece valutati non interessanti per l'utente. Infine, si devono considerare anche le implicazioni dovute alla generazione dinamica dei contenuti unite alla natura dei contatti opportunistici: quando due agenti si incontrano (se si incontrano), non è possibile assumere che

un dato contenuto sia già stato generato. È plausibile supporre, quindi, che a un agente non vengano mai proposti dei contenuti che sono invece presenti nel suo rispettivo resource vector globale.

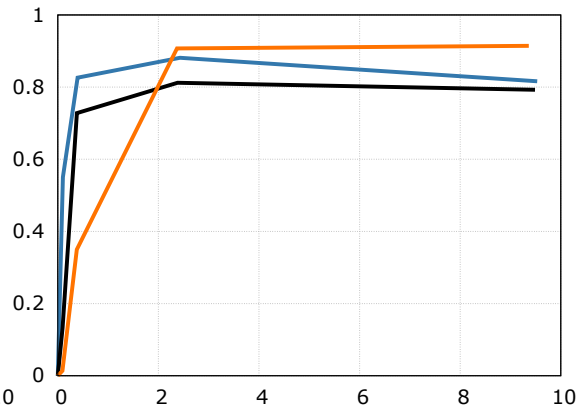


(a) $buffer\_size = 5$

(b) $buffer\_size = 10$

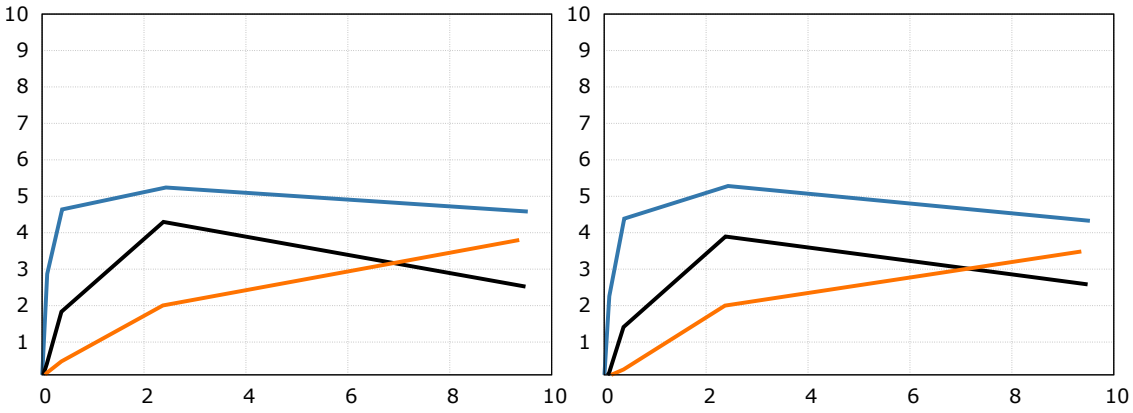


(c) $buffer\_size = 25$



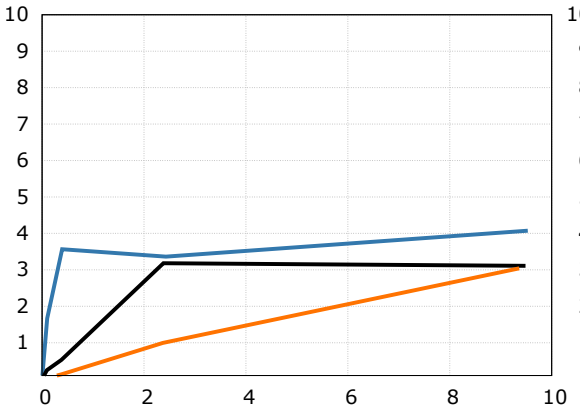
(d) $buffer\_size = 50$

Figura 6.11: Similarità tra la media dei grafi locali degli agenti e il grafo di conoscenza globale al variare del numero di contatti e dimensione del buffer di suggerimento. In azzurro è riportato l'andamento per 50 agenti, in nero quello per 100 e in arancio quello per 200.

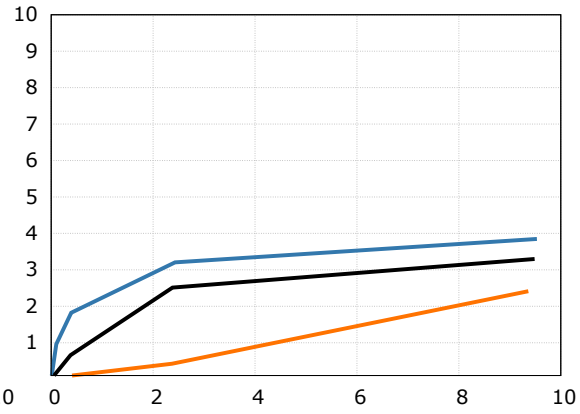


(a) $buffer\_size = 5$

(b) $buffer\_size = 10$



(c) $buffer\_size = 25$



(d) $buffer\_size = 50$

Figura 6.12: Percentuale media degli item presenti nel resource vector globale che sono stati scaricati dagli agenti alla fine delle simulazioni, al variare del numero di contatti e dimensione del buffer di suggerimento. In azzurro è riportato l'andamento per 50 agenti, in nero quello per 100 e in arancio quello per 200.

6.5.4 Adattabilità del buffer di suggerimento

In questa sezione vengono riportati i grafici relativi ai buffer di suggerimento di tre utenti casualmente estratti dalle precedenti simulazioni (Sezione 6.5.3). Più precisamente, ciascun grafico rappresenta l'andamento della media dei contenuti presenti nei buffer durante le simulazioni al variare della dimensione dello stesso.

È interessante notare come, per dimensioni del buffer superiori a 1, la media dei contenuti presenti al suo interno tenda a risultare più stabile rispetto al caso in cui venga scaricato qualsiasi item considerato interessante da PLIERS (con valore diverso da 0). Tale meccanismo risulta, quindi, adattarsi meglio ai contenuti scoperti durante i contatti opportunistici, suggerendo all'utente il download dei soli item il cui valore supera la media di quelli precedentemente esaminati da PLIERS.

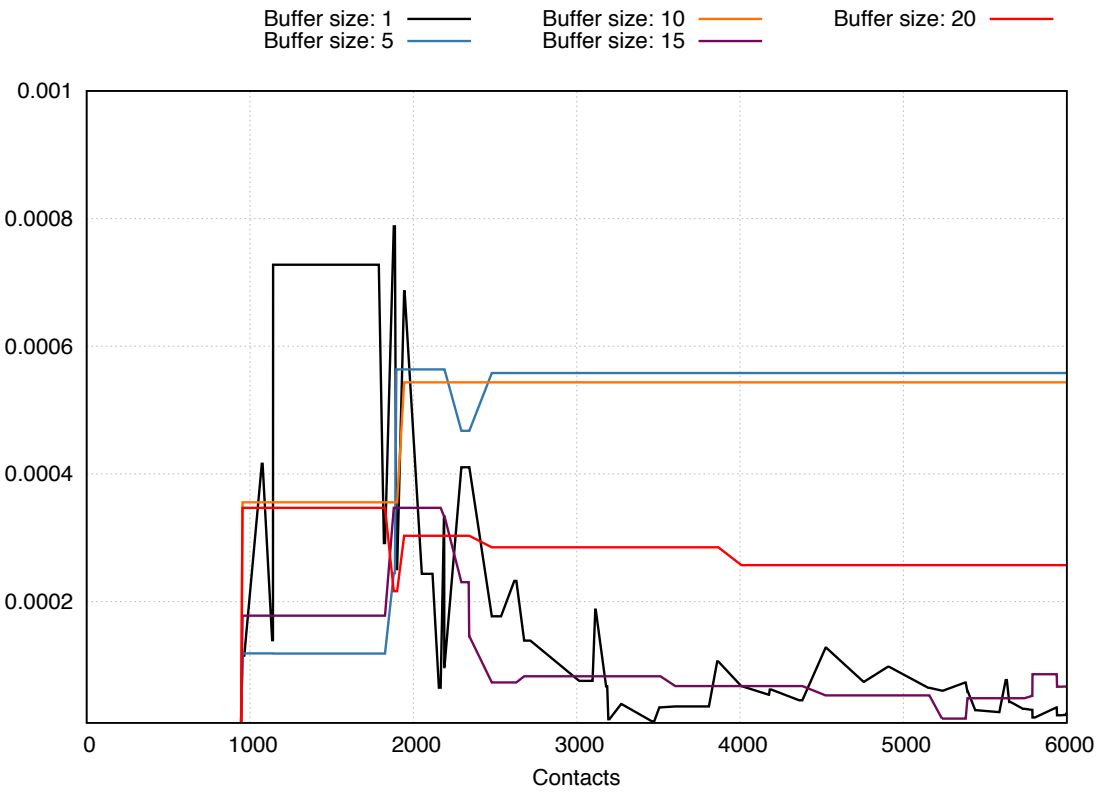


Figura 6.13: Andamento del buffer per diverse dimensioni.

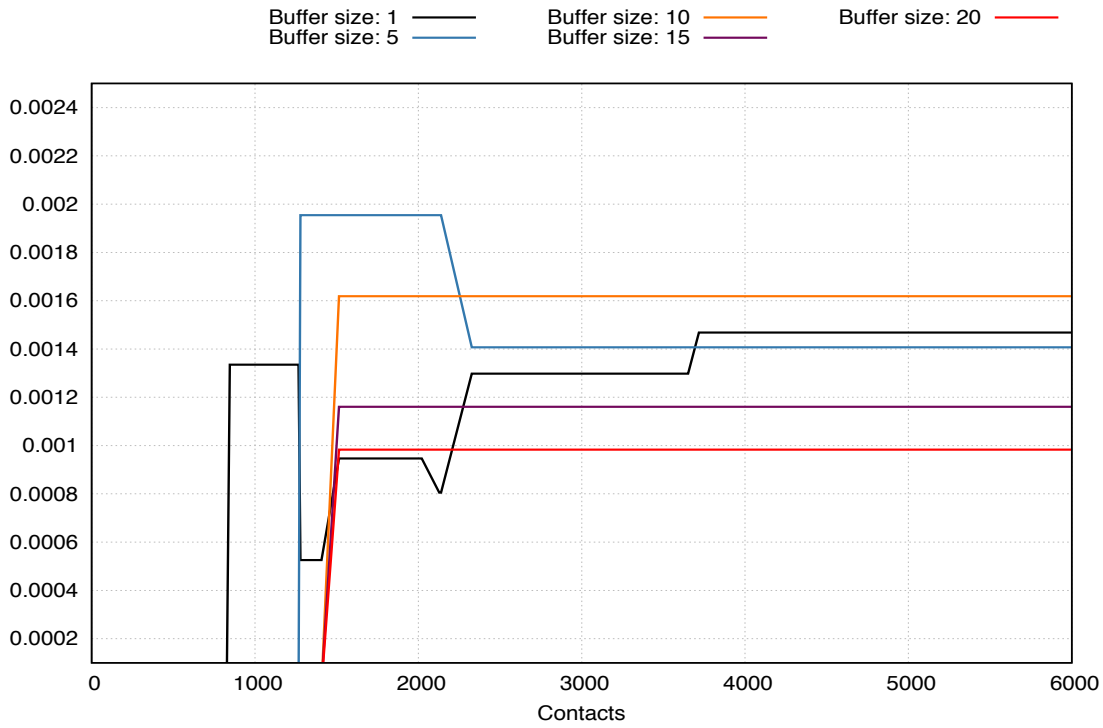


Figura 6.14: Andamento del buffer per diverse dimensioni.

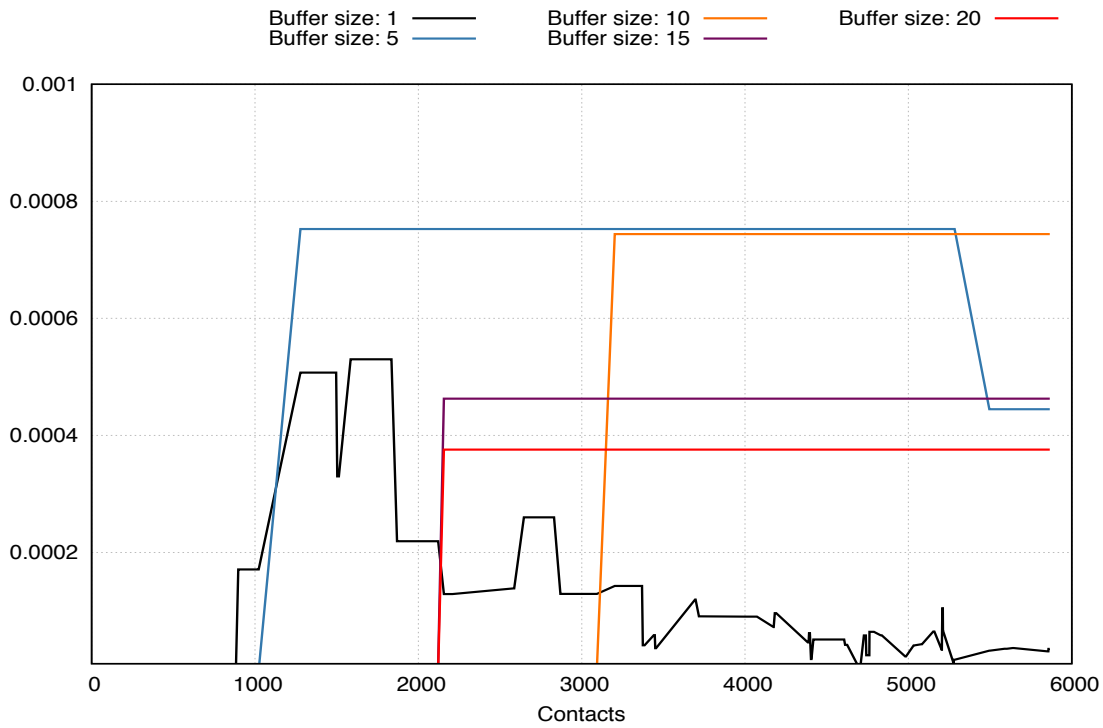


Figura 6.15: Andamento del buffer per diverse dimensioni.

Capitolo 7

Conclusioni

Lo scopo di questo lavoro di tesi era quello di definire un algoritmo di context-reasoning per dispositivi mobili e, nella fattispecie, per il middleware CAMEO, in uno scenario in cui i nodi (utenti/dispositivi) si scambiano informazioni a vicenda sfruttando le opportunità offerte dai contatti opportunistici.

L'algoritmo, basandosi su informazioni di contesto relative a contenuti e/o risorse disponibili in rete, e gli interessi di uno o più utenti, doveva essere in grado di calcolare l'utilità dei contenuti (o risorse) per il nodo locale in base ad una metrica di interesse. Quindi, il fine ultimo di tale algoritmo, consisteva nell'ottimizzare la distribuzione di contenuti in una rete opportunistica, basandosi sulle informazioni di contesto degli utenti.

In questo lavoro di tesi sono stati studiati gli algoritmi di context-reasoning per sistemi mobili e pervasivi che rappresentano il corrente stato dell'arte. Dal momento che in CAMEO tutti i contenuti vengono associati a dei tag (generati automaticamente dalle applicazioni o creati dagli utenti), lo studio per definire un algoritmo di context-reasoning, orientato alla distribuzione dei contenuti, si è focalizzato sull'analisi di quegli algoritmi che, in letteratura, vengono indicati con il nome di Tag-Based Recommender System.

La soluzione proposta, l'algoritmo PLIERS (PopuLarity-based ItEm Recommender System), utilizza un approccio basato su grafi (bipartiti e tripartiti) per modellare le relazioni che legano tra loro gli utenti, i contenuti da essi scaricati

e/o generati e i tag associati a questi ultimi. Questo tipo di grafo viene definito come *"grafo di conoscenza"*. La novità del lavoro consiste nella definizione di un nuovo algoritmo per tag-based recommender system che migliora le prestazioni, rispetto allo stato dell'arte, in termini di precisione nella selezione dei contenuti da suggerire all'utente locale basandosi sui suoi interessi. Inoltre, tale approccio, è stato applicato per la prima volta nelle reti opportunistiche come algoritmo di reasoning per la realizzazione di servizi di condivisione di contenuti e/o risorse context- e social-aware.

Per validare la soluzione proposta, è stato utilizzato un dataset derivato dalla rete sociale Twitter, contenente le attività (tweet e re-tweet) di 1630594 utenti in un arco temporale di circa due anni (2012-2014). Sono state confrontate le raccomandazioni effettuate, su un campione di 5000 utenti, da parte di PLIERS con i risultati ottenuti da altri algoritmi presenti in letteratura: ProbS, HeatS e una soluzione ibrida ProbS+HeatS. In questo modo è stato dimostrato come PLIERS selezioni i contenuti da suggerire privilegiando quelli con popolarità simile a quella degli item già in possesso dell'utente target, al contrario degli altri algoritmi che, invece, prediligono contenuti con massima o minima popolarità rispetto agli interessi dell'utente di riferimento.

Allo scopo di valutare l'efficiacia di PLIERS in ambiente opportunistico, è stato realizzato un simulatore che, utilizzando delle tracce di mobilità, ricrei il comportamento atteso da parte dei nodi nel momento in cui si verifica un contatto ravvicinato tra essi (contatto opportunistico).

In ambiente opportunistico, il grafo che rappresenta la relazione tra un utente ed i contenuti che possiede o genera nel tempo rappresenta un'ulteriore informazione di contesto che viene scambiata tra i nodi quando si incontrano. Ogni nodo, quindi, mantiene solo una visione parziale del grafo di conoscenza (*"grafo di conoscenza locale"*) e, ad ogni contatto, scambia questo grafo con i propri vicini, cercando di ricostruire una visione globale delle informazioni in rete (*"grafo di conoscenza globale"*).

Tali simulazioni hanno permesso di verificare quanto il grafo di conoscenza costruito da ciascun nodo differisca da quello globale, costruito sulla base del

dataset a disposizione. In questo modo è stato possibile, inoltre, scoprire quanti contatti opportunistici siano necessari prima che un nodo riesca ad ottenere una buona approssimazione dell'intero grafo di conoscenza. Infine, simulando la generazione dinamica dei contenuti nella rete, è stato possibile misurare la precisione con cui i contenuti vengano selezionati rispetto alla visione globale del grafo di conoscenza.

7.1 Sviluppi futuri

Sarebbe interessante utilizzare diversi tipi di dataset per confrontare, dal punto di vista semantico, le raccomandazioni effettuate da PLIERS con quelle proposte dagli altri algoritmi considerati. Infatti, seppur l'insieme dei dati estratti da Twitter rappresenti molto bene le relazioni sociali tra gli utenti, nel dataset utilizzato è stata rilevata una presenza notevole di tag non significativi per una chiara descrizione semantica dei contenuti. Come è stato spiegato nel Capitolo 4.1, questi tag vengono tipicamente utilizzati dagli utenti solo per aumentare la propria visibilità all'interno della rete sociale e, dal momento che rendono difficile (se non impossibile) un confronto semantico tra i tag inizialmente in possesso degli utenti e quelli proposti dagli algoritmi, vengono considerati come "rumore". Alcuni dei dataset che potrebbero essere utilizzati per condurre ulteriori valutazioni sono:

MovieLens Un dataset contenente informazioni estratte dal social network *movielens.org*. Contiene i dati riguardanti film (titolo, copertina, trama, genere, etc...) e la lista di tag associati a ciascuno di essi da parte degli utenti del sito.

BibSonomy Contenente i tag assegnati dagli utenti a pubblicazioni scientifiche tramite il sito *bibsonomy.org*.

Delicious Il sito *delicious.com* permette ai propri utenti di condividere link di pagine web a cui è possibile associare dei tag per descriverne il contenuto.

Un ulteriore sviluppo futuro potrebbe consistere nell'implementazione, all'interno del simulatore opportunistico, dei metodi proposti nel Capitolo 5.3.2 per il

controllo del grafo di conoscenza locale. In questo modo, sarebbe possibile studiare come varia l'accuratezza delle proposte effettuate da PLIERS nel caso in cui venga limitata la dimensione massima del grafo locale. Questo permetterebbe, infine, di realizzare il prototipo di un'applicazione per dispositivi mobili allo scopo di testare l'effettiva efficacia dell'algoritmo in una situazione reale. Tramite un'applicazione di questo genere sarebbe possibile, infatti, verificare quanto i contenuti proposti dall'algoritmo siano effettivamente interessanti per l'utente finale utilizzando un meccanismo di feedback automatizzato: nel caso in cui l'utente scaricasse un contenuto proposto da PLIERS sarebbe considerato come un feedback positivo, altrimenti, se viene scartato, come un feedback negativo.

Bibliografia

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. A. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *Center for Embedded Network Sensing*, 2007.

[2] A. Agostini, C. Bettini, and D. Riboni. A performance evaluation of ontology-based context reasoning (experience report). In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications-Workshops (PerCom Workshops 2007)*, pages 19–23, 2007.

[3] K. Ali and W. Van Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 394–401. ACM, 2004.

[4] L. A. N. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the national academy of sciences*, 97(21):11149–11152, 2000.

[5] V. Arnaboldi, M. Conti, and F. Delmastro. Cameo software architecture and apis: Technical specification.

[6] V. Arnaboldi, M. Conti, and F. Delmastro. Cameo: A novel context-aware middleware for opportunistic mobile social networks. *Pervasive and Mobile Computing*, 11:148–167, 2014.

[7] V. Arnaboldi, M. Conti, F. Delmastro, G. Minutiello, and L. Ricci. Sensor mobile enablement (sme): A light-weight standard for opportunistic sensing

services. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 236–241. IEEE, 2013.

[8] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[9] S. Berg. Snowball sampling—i. *Encyclopedia of statistical sciences*, 1988.

[10] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180, 2010.

[11] P. Blunsom. Hidden markov models. *Lecture notes, August*, 15:18–19, 2004.

[12] C. Boldrini, M. Conti, F. Delmastro, and A. Passarella. Context-and social-aware middleware for opportunistic networks. *Journal of Network and Computer Applications*, 33(5):525–541, 2010.

[13] C. Boldrini, M. Conti, and A. Passarella. Users mobility models for opportunistic networks: the role of physical locations. *Proc. of IEEE WRECOM*, page 23, 2007.

[14] C. Boldrini, M. Conti, and A. Passarella. Exploiting users' social relations to forward data in opportunistic networks: The hibop solution. *Pervasive and Mobile Computing*, 4(5):633–657, 2008.

[15] C. Boldrini and A. Passarella. Hcmm: Modelling spatial and temporal properties of human mobility driven by users' social relationships. *Computer Communications*, 33(9):1056–1074, 2010.

[16] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[17] B. Burns, O. Brock, and B. N. Levine. Mora routing and capacity building in disruption-tolerant networks. *Ad hoc networks*, 6(4):600–620, 2008.

[18] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. Technical report, RFC 4838, April, 2007.

[19] A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[20] M. Conti, S. K. Das, C. Bisdikian, M. Kumar, L. M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, and F. Zambonelli. Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence. *Pervasive and Mobile Computing*, 8(1):2–21, 2012.

[21] M. Conti and M. Kumar. Opportunities in opportunistic computing. *Computer*, (1):42–50, 2010.

[22] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

[23] M. J. Demmer. *A delay tolerant networking and system architecture for developing regions*. PhD thesis, Citeseer, 2008.

[24] M. K. Denko. *Mobile Opportunistic Networks: Architectures, Protocols and Applications*. CRC Press, 2011.

[25] F. Giannotti, D. Pedreschi, A. Pentland, P. Lukowicz, D. Kossmann, J. Crowley, and D. Helbing. A planetary nervous system for social mining and collective awareness. *European Physical Journal-Special Topics*, 214(1):49, 2012.

[26] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[27] D. Goodman, J. Borras, N. B. Mandayam, and R. D. Yates. Infostations: A new system model for data and messaging services. In *Vehicular Technology Conference, 1997, IEEE 47th*, volume 2, pages 969–973. IEEE, 1997.

[28] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 33–37. IEEE, 2004.

[29] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and mobile computing*, 2(1):37–64, 2006.

[30] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In *Pervasive Computing*, pages 167–180. Springer, 2002.

[31] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[32] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79, 2004.

[33] T. Hossmann, T. Spyropoulos, and F. Legendre. Know thy neighbor: Towards optimal mapping of contacts to social graphs for dtn routing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[34] S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications*, 11(3):327–339, 2006.

[35] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity technical report. Technical report, 2001.

[36] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[37] C. Keßler, M. Raubal, and C. Wosniok. Semantic rules for context-aware geographical information retrieval. In *Smart Sensing and Context*, pages 77–92. Springer, 2009.

[38] A. Kiran, A. Patil, and S. Limkar. Survey of routing techniques in opportunistic networks.

[39] B. Y. Lim and A. K. Dey. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 13–22. ACM, 2010.

[40] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[41] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003.

[42] C. Liu and W.-X. Zhou. An improved heats+probs hybrid recommendation algorithm based on heterogeneous initial resource configurations. *CoRR*, abs/1005.3124, 2010.

[43] A. Mathes. Folksonomies-cooperative classification and communication through shared metadata, 2004.

[44] M. Mikalsen and A. Kofod-Petersen. Representing and reasoning about context in a mobile environment. In *Proceedings of the First International Workshop on Modeling and Retrieval of Context. CEUR Workshop Proceedings*, volume 114, pages 25–35, 2004.

[45] B. Motik, I. Horrocks, and S. M. Kim. Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 63–72. ACM, 2012.

[46] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *World of wireless mobile and multimedia networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pages 183–189. IEEE, 2005.

[47] M. Musolesi and C. Mascolo. Designing mobility models based on social network theory. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(3):59–70, 2007.

[48] G. J. Nalepa and S. Bobek. Rule-based solution for context-aware reasoning on mobile devices. *Computer Science and Information Systems*, 11(1):171–193, 2014.

[49] A. Padovitz, S. W. Loke, and A. Zaslavsky. The ecora framework: A hybrid architecture for context-oriented pervasive computing. *Pervasive and mobile computing*, 4(2):182–215, 2008.

[50] L. Peizhi and Z. Jian. A context-aware application infrastructure with reasoning mechanism based on dempster-shafer evidence theory. In *Vehicular*

Technology Conference, 2008. VTC Spring 2008. IEEE, pages 2834–2838. IEEE, 2008.

[51] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134–141, 2006.

[52] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[53] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2009.

[54] D. Riboni and C. Bettini. Context-aware activity recognition through a combination of ontological and statistical reasoning. In *Ubiquitous Intelligence and Computing*, pages 39–53. Springer, 2009.

[55] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. In *Applications of Data Mining to Electronic Commerce*, pages 115–153. Springer, 2001.

[56] A. Sinner and T. Kleemann. Krhyper–in your pocket. In *Automated Deduction–CADE-20*, pages 452–457. Springer, 2005.

[57] T. Small and Z. J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 233–244. ACM, 2003.

[58] K. Teymourian, O. Streibel, A. Paschke, R. Alnemr, and C. Meinel. Towards semantic event-driven systems. In *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, pages 1–6. IEEE, 2009.

[59] A. Vahdat, D. Becker, et al. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000.

[60] M. Weiser. Reasoning about uncertain contexts in pervasive computing environments.

[61] Y.-C. Zhang, M. Blattner, and Y.-K. Yu. Heat conduction process on community networks as a recommendation model. *Physical review letters*, 99(15):154301, 2007.

[62] Z.-K. Zhang, T. Zhou, and Y.-C. Zhang. Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. *Physica A: Statistical Mechanics and its Applications*, 389(1):179–186, 2010.

[63] Z.-K. Zhang, T. Zhou, and Y.-C. Zhang. Tag-aware recommender systems: a state-of-the-art survey. *Journal of computer science and technology*, 26(5):767–777, 2011.

[64] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

[65] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.

[66] X. Zhou, X. Tang, X. Yuan, and D. Chen. Spbca: Semantic pattern-based context-aware middleware. In *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, pages 891–895. IEEE, 2009.