UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi Di Laurea

Tecniche Di Raccomandazione Sociale Di Beni Artistici Basate Su Linked Open Data

Laureando

Alessio De Angelis

Matricola 417021

Relatore

Prof. Alessandro Micarelli

Correlatore

Prof. Giuseppe Sansonetti

Anno Accademico 2013/2014

Alla mia famiglia e a chi ha sempre creduto in me

"Sono convinto che l'Informatica abbia molto in comune con la Fisica. Entrambe si occupano di come funziona il mondo a un livello abbastanza fondamentale. La differenza, naturalmente, è che mentre in Fisica devi capire come è fatto il mondo, nell'Informatica sei tu a crearlo. Dentro i confini del computer, sei tu il creatore. Controlli, almeno potenzialmente, tutto ciò che vi succede. Se sei abbastanza bravo, puoi essere un dio. Su piccola scala."

Linus Torvalds

"Non bisogna essere stanchi di indagare se ciò che cerchiamo è l'eccellenza."

Marco Tullio Cicerone

"Guida tu stesso la tua canoa[...]. Incontrerai sulla tua rotta difficoltà e pericoli, banchi e tempeste. Ma senza avventura, la vita sarebbe terribilmente monotona. Se saprai manovrare con cura, navigando con lealtà e gioiosa persistenza, non c'è ragione perché il tuo viaggio non debba essere un completo successo; poco importa quanto piccolo fosse il ruscello dal quale un giorno partisti."

Robert Baden Powell

"L'arte è capace di esprimere e rendere visibile il bisogno dell'uomo di andare oltre ciò che si vede, manifesta la sete e la ricerca dell'infinito. Anzi, è come una porta aperta verso l'infinito, verso una bellezza e una verità che vanno al di là del quotidiano. E un'opera d'arte può aprire gli occhi della mente e del cuore, sospingendoci verso l'alto."

Papa Benedetto XVI

"L'arte oltrepassa i limiti nei quali il tempo vorrebbe comprimerla ed indica il contenuto del futuro."

Vasilij Kandinskij

Ringraziamenti

Questa tesi è la conclusione del mio ciclo di studi, sicuramente impegnativo ma decisamente appagante e formativo. Mi ha permesso, tramite il progetto Erasmus, di maturare un'esperienza di inestimabile valore, entrando a contatto con numerose culture e modi di vivere ed allargando così la mia visione che ho del mondo.

Vorrei innanzitutto ringraziare tutta la mia famiglia che, in tutti questi anni, mi ha sempre sostenuto e creduto in me, dandomi forza e sicurezza. Siete ciò che ho di più importante e che sempre avrò. Scegliendo lo scenario delle raccomandazioni in ambito del patrimonio artistico e culturale, ho immediatamente pensato a mamma e papà, voi così tanto appassionati di arte, sempre in giro ad ammirare le meraviglie che il Bel Paese ci può offrire!

Un ringraziamento particolare va al mio relatore e correlatore, il Prof. Alessandro Micarelli e il Prof. Giuseppe Sansonetti che, in modo paziente e disponibile, mi hanno seguito nella stesura di questa tesi, fornendomi ottimi consigli e raccomandazioni (sempre per rimanere in tema). Spero di poter continuare il bellissimo rapporto che si è instaurato in questi mesi, se non a livello lavorativo, sicuramente a livello umano.

Voglio ringraziare tutti i miei amici e compagni di viaggio di questa avventura che è la vita, dal primo all'ultimo, in qualsiasi parte del mondo ora vi troviate, dall'Italia alla Norvegia, dalla Spagna alla Corea, dalla Polonia all'India per ricordarne alcune.

Un sentito ringraziamento va anche a tutte le maestre e i professori che ho cono-

sciuto durante il mio intero periodo da studente, partendo dalle scuola elementare: avete saputo far crescere in me la passione per il sapere, per lo scoprire sempre il perchè delle cose, andando ogni volta a fondo nella conoscenza, senza fermarsi mai in superficie.

Introduzione

Ogni giorno ciascun individuo assume decisioni sulla base di suggerimenti di diverso tipo e provenienza: dialoghi, lettere, articoli di giornale, riviste, guide di viaggio, libri, radio, televisione, pubblicità e via discorrendo. Tali consigli sono ritenuti tanto più affidabili quanto maggiore è il grado di conoscenza della sorgente da parte dell'individuo.

I *Sistemi di Raccomandazione (Recommender System, RS)* nascono con l'obiettivo di identificare oggetti di possibile interesse per uno specifico utente. Gli ambiti sono numerosi e variegati: spaziano dal brano musicale da ascoltare durante un viaggio in auto al ristorante dove cenare con i familiari, da un film da vedere al cinema con gli amici al museo da visitare in una città straniera.

Con milioni di utenti distribuiti su tutto il pianeta, i social network quali Facebook e Twitter sono diventati, nel corso degli ultimi anni, alcune tra le applicazioni più visitate e popolari nel Web. La mole di dati generata dagli utenti dei social network è tale che per essi è più appropriato usare il termine di *Big Data*. Lo sviluppo delle tecnologie wireless e di acquisizione della posizione di un dispositivo mobile ha permesso, inoltre, di esplorare nuove frontiere. Ad esempio, l'utente può condividere online le coordinate geospaziali del museo che sta visitando e commentarlo. Le tecniche per modellare gli utenti sulla base della loro attività nei social network stanno così destando una sempre maggiore attenzione.

Un problema di cui tuttavia soffrono questi servizi è l'assenza di uno standard con cui elaborare e gestire in modo uniforme le informazioni sociali. Le tecnologie del *Semantic Web* possono quindi essere adottate per strutturare e ar-

ricchire i dati sociali in modo da integrarli, dotarli di una semantica e renderli così processabili all'interno della piattaforma. I *Linked Open Data (LOD)* derivano dall'interpretare il Web attuale come *Web of Data*, ovverosia un enorme repository di documenti strutturati interconnessi tra loro sulla base del loro significato. Tali dati devono essere interrogabili e processabili liberamente da chiunque, senza limite alcuno. Hearth e Bizer sintetizzano così il paradigma del Web of Data [HB11]:

> "*Linked Data provides a publishing paradigm in which not only documents, but also data, can be a first class citizen of the Web, thereby enabling the extension of the Web with a global data space based on open standards - the Web of Data.*"

Oggetto di questa tesi è *Cicero*, un sistema di raccomandazione chiamato così in onore di Marco Tullio Cicerone, uno dei personaggi più influenti ai tempi dell'antica repubblica romana, il cui nome è diventato, a partire dal XVIII secolo, il titolo che si attribuisce per antonomasia alle guide che conducono i turisti in visita artistica a Roma. Questo sistema nasce con l'intento di analizzare come, all'interno di un processo di raccomandazione sociale, i dati estratti dalle attività di un utente su un social network possano essere arricchiti con le basi di conoscenza semantica fornite dai LOD. In particolare, la raccomandazione è applicata al dominio del patrimonio artistico e culturale mondiale. A questo scopo, il framework inferisce le preferenze utente dal social network Facebook, in termini di luoghi visitati o menzionati nello *status*, combinandole con la conoscenza semantica propria del Web of Data. Le sorgenti LOD adoperate per espandere le informazioni sociali sono *Europeana* e *DBpedia*. La prima è una biblioteca digitale online che espone liberamente opere dei beni culturali europei, come documenti, foto di dipinti e file musicali, raccolti grazie al contributo di oltre 2000 istituzioni europee che hanno deciso di aderire al progetto. Il secondo dataset prende il nome dal progetto che ha come fine l'estrazione delle informazioni strutturate di

Wikipedia e il loro rilascio sul Web come Linked Open Data in formato *Resource Description Framework*[1] *(RDF).* Questi è uno standard proposto da W3C[2] per la codifica, lo scambio e il riutilizzo di metadati strutturati.

Obiettivo del lavoro presentato in questa dissertazione è analizzare quanto le informazioni semantiche appartenenti a un grafo LOD possano influenzare positivamente le prestazioni di un Social Recommender System. A tal proposito, vengono proposti vari algoritmi di raccomandazione che implementano tecniche allo Stato dell'Arte per la modellazione utente su grafi. Alcuni di essi sono inoltre arricchiti con le risorse estratte dai Semantic Web. I test sperimentali condotti testimoniano i vantaggi dell'approccio adottato: si mostra, infatti, come la qualità della raccomandazione applicata a dataset reali, estratti monitorando l'attività di utenti Facebook, migliori in presenza dei Linked Open Data durante l'elaborazione. La valutazione si avvale del contributo di un campione costituito da 25 sperimentatori che hanno partecipato ai test su base volontaria. I loro giudizi, espressi in un range di interi compresi tra 1 e 5 secondo i principi della scala di Likert a 5 valori, sono stati dapprima esaminati dal punto di vista prestazionale in termini di *Normalized Cumulative Discounted Gain* e, successivamente, verificati mediante un test di significatività statistica, l'*ANalisys Of VAriance (ANOVA)*, con l'obiettivo di scoprire se le differenze riscontrate siano dovute al caso o a proprietà intrinseche dei vari componenti interni degli algoritmi.

Il presente elaborato di tesi è strutturato come segue.

Il **Capitolo 1** riporta una panoramica sul campo di ricerca dei sistemi di raccomandazione, soffermandosi in particolar modo su due aspetti. Il primo riguarda le modalità con cui l'esplorazione del Social Web può attivamente contribuire alle operazioni di modellazione utente. Il secondo, invece, è relativo ai Location-Based Social Network (LBSN), social network nati principalmente con l'idea di consen-

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

tire agli utenti la condivisione online, con la propria comunità virtuale, delle informazioni relative alla propria posizione geografica corrente. Questa funzionalità ha sicuramente tratto giovamento dalla crescente diffusione dei dispositivi mobili con strumenti di GPS integrati. Inoltre, nel capitolo vengono illustrate le principali tecniche di raccomandazione proposte in letteratura.

Il **Capitolo 2** passa in rassegna il vasto mondo dei Linked Open Data, approfondendo, tra gli altri, i concetti di annotazioni semantiche, di ontologie, di RDF, di SPARQL (il linguaggio dichiarativo sviluppato per consentire agli utenti di formulare query ed estrarre informazioni dalle basi di conoscenza distribuite sul Web of Data). Sono inoltre descritti in dettaglio i progetti LOD DBpedia ed Europeana, nonchè Neo4j, un Database Management System NoSQL appartenente allo scenario dei graph db, in quanto impiegati nello sviluppo di Cicero.

Il **Capitolo 3** presenta lo Stato dell'Arte dei Recommender System basati sui dati del Social Web e del Linked Open Data. Sono qui esposti i vantaggi e gli svantaggi riconosciuti di ciascuno di essi, in relazione agli algoritmi proposti in Cicero.

Il **Capitolo 4** illustra i fondamenti teorici del sistema. Nello specifico, si analizzano le varie fasi che permettono al framework di modellare le preferenze dell'utente e di produrre suggerimenti di beni artistici e culturali che possano essere in linea con i suoi gusti. Pertanto, vengono approfondite le tecniche di estrazione dei check-in dalle attività degli account sul social network Facebook e la connessa operazione di disambiguazione dei punti di interesse (Point Of Interest, POI) trovati. Vengono esposti anche i diversi algoritmi di filtraggio adottati in Cicero e l'arricchimento semantico dei dati sociali ottenuto in seguito ad un'esplorazione di Europeana e DBpedia.

Il **Capitolo 5** descrive i dettagli dell'architettura software multi-tier di Cicero, analizzandola principalmente dal punto di vista della Logic View. La trattazione si sofferma sui Driver Architetturali Significativi (ASR), sulle tattiche architetturali e sui pattern di design e architetturali.

Il **Capitolo 6** illustra le prove sperimentali condotte su Cicero, al fine di verificarne l'efficacia in termini prestazionali. La valutazione segue un approccio *user-centric*, incentrato cioè sulle percezioni che l'utente ha del sistema durante l'interazione con esso. La sperimentazione si focalizza altresì sulle metriche di accuratezza percepita, serendipità, novità e diversità. Varie sequenze di algoritmi sono dapprima confrontate tra loro in termini di Normalized Discounted Cumulative Gain (NDCG). I risultati così conseguiti sono successivamente verificati con l'ANOVA Test per garantirne una valenza statistica.

Il **Capitolo 7**, infine, espone le conclusioni e i possibili sviluppi futuri delle attività di ricerca avviate con l'ideazione e la realizzazione del sistema oggetto del lavoro di tesi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Tecniche di raccomandazione

La branca di ricerca dei Recommender System (RS) vede i suoi albori ad inizio anni Novanta, periodo in cui anche il Web stava iniziando a muovere i suoi primi passi. Essa si occupa dello sviluppo di tool e tecniche software in grado di fornire suggerimenti di oggetti considerati utili per una data persona [BLPR12]. I sistemi di raccomandazione collezionano infatti, in modo proattivo, informazioni sulle preferenze dei loro utilizzatori per un ben determinato dominio di elementi: esempi sono brani musicali, libri, film, punti di interesse (POI), beni artistici e culturali, destinazione di viaggi, materiale didattico. L'estrazione di tali dati avviene in maniera implicita e trasparente all'utente, dopo un'analisi che coinvolge tipicamente lo storico delle loro attività (canzoni ascoltate, prodotti visualizzati, luoghi geografici in cui si è effettuato un checkin) oppure esplicitamente. Alcuni servizi collezionano ad esempio i giudizi e le recensioni dei propri fruitori relativi alla merce acquistata o ai film visti.

Nel processo di raccomandazione gli algoritmi a volte esaminano anche le caratteristiche demografiche degli utenti, quali età, sesso, nazionalità, città in cui si vive, senza escludere le informazioni sociali, quali bacheca, profilo, amici che si seguono in un social network. Diffuso è anche l'uso dell'*Internet of Things* come sorgente dati: basti pensare alle numerose applicazioni che sfruttano le coordinate GPS. Su questa funzionalità, ad esempio, i Location-Based Social

Network (LBSN) basano le proprie attività [BOHG13].

I benefici derivati dall'impiego di algoritmi di raccomandazione sono duplici. Da un lato aiutano gli utenti, dall'altro sono potenti alleati di molti business e prodotti commerciali, in primis la pubblicità e i siti di e-commerce. Si parte infatti dal successo di Amazon.com, che è stato in un certo senso il pionere dei Recommender System, fino ad arrivare ad altri attori importanti del panorama commerciale, quali Youtube, Spotify, Pandora Radio, Last.FM, Ebay.

I vantaggi per l'utente includono un'efficienza maggiore nel trovare gli articoli prediletti, maggiore confidenza nelle scelte di acquisto e la possibilità di scoprire qualcosa di nuovo in un determinato dominio. Per i venditori questa tecnologia può invece aumentare significativamente la probabilità che le persone comprino effettivamente gli articoli loro suggeriti; ciò incrementa anche la soddisfazione e la lealtà degli utenti in merito alla piattaforma su cui si è acquistato il prodotto.

I sistemi di raccomandazione cercano inoltre di bilanciare vari fattori quali l'accuratezza, la precizione, la novità, la serendipità, la diversità dei risultati.

1.1 Social Web

Il Social Web è una miniera d'oro per sviluppatori e ricercatori delle tecniche di modellazione utente che studiano come le tracce lasciate dall'utente nel Web, come i click, i voti, le risorse condivise possano essere trasformate in rappresentazioni vantaggiose per una data applicazione. Ad esempio, i Tweet che le persone pubblicano su Twitter possono essere sfruttati in vari contesti, come la raccomandazione personalizzata di news. Analizzando il feed dello status degli utenti di Twitter, possono essere consigliate foto, per esempio, come avviene in [AGHT11]. Il discorso può essere allargato ad altri servizi, Facebook in primis.

Per applicare la modellazione utente in tali scenari, è essenziale tuttavia la comprensione della semantica dei messaggi condivisi sulle piattaforme sociali.

Inoltre, come esposto in [AGHT07], importanti scoperte scientifiche in sociologia e psicologia sono incentrate sull'influenza dei social network nelle dinamiche che determinano i gusti delle persone, le loro preferenze e attività. Un esempio è il principio dell'*omofilia*, ampiamente riconosciuto nella letteratura dei sistemi di raccomandazione. McPherson et al. hanno enunciato in [MSLC01] che *"Similarity breeds connections"*, cioè che la similarità permette di attivare connessioni. Gli autori dell'articolo hanno infatti dimostrato come le reti di amicizia delle persone siano omogenee in termini di numerose caratteristiche sociodemografiche, comportamentali e nei rapporti interpersonali. In altre parole, convidiamo molti attribuiti con le persone a noi più strette. L'omofilia comporta pertanto la tendenza alla creazione di gruppi omogenei di persone che si rispecchiano letteralmente le une nelle altre. Invertendo questo principio si può ragionare sul fatto che, possedendo informazioni sulle relazioni all'interno della rete personale di ciascun individuo, possono essere inferite alcuni loro attributi, come i gusti in termini di opere d'arte preferite, ad esempio.

E' probabile che alcune delle similarità presenti nella rete di conoscenza siano causate dall'influenza e dalle interazioni tra le persone al suo interno. Infatti, quando è necessario effettuare una scelta, ad esempio selezionare il modello di cellulare o di macchina da acquistare e che rispecchi maggiormente le proprie esigenze, si è spesso propensi a fidarsi maggiormente dei consigli di un amico stretto rispetto a delle recensioni redatte da uno sconosciuto, seppur se questi sia magari un addetto ai lavori in quell'ambito.

A partire da queste considerazioni, emerge pertanto come il Social Web risulti un valido strumento nelle dinamiche di profilazione utente e ne giustifica una reputazione sempre maggiore nella letteratura dei sistemi di raccomandazione.

1.2 Location-Based Social Network

Nel panorama dei social network, un'alta popolarità è stata conseguita dai Location-Based Social Network (LBSN). Con questo termine si intendono quelle piattaforme sociali che permettono ai propri utenti di tracciare, annotare, condividere le esperienze relative ai posti visitati durante la vita quotidiana. Gli utenti possono così notificare agli amici le proprie coordinate GPS, collegandole non solo ai messaggi pubblicati nelle bacheche personali ma anche ad altri contenuti, come foto o video. Viene pertanto lasciata una traccia digitale dei propri movimenti.

Un individuo con un dispositivo mobile come smartphone o tablet potrebbe commentare sul proprio profilo sul suo social media preferito un museo che ha appena visitato con un *check-in*, annotandolo cioè con la rispettiva latitudine e longitudine. Altri utenti potrebbero essere invece interessati ad allargare la propria comunità sociale grazie a suggerimenti forniti analizzando storici di posizioni comuni: persone che costantemente scalano la stessa montagna e non si conoscono potrebbero, ad esempio, essere messe in contatto.

Esistono LBSN nativi, creati cioè espressamente con l'ottica di garantire questa funzionalità. Esemplare principe è Foursquare[1]. Altri importanti social network come Facebook o Twitter, apprezzando le potenzialità che derivano dal consentire all'utente di pubblicare online la propria localizzazione fisica, hanno deciso in un secondo momento di includere questo servizio tra quelli disponibili.

Zheng et al. in [ZZM+11] definiscono formalmente la natura di un LBSN.

> "*A location-based social network (LBSN) does not only mean adding a location to an existing social network so that people in the social structure can share location-embedded information, but also consists of the new social structure made up of individuals connected by the interdependency derived from their locations in the physical world as well as their location-tagged media content, such as photos, video, and text.*

[1] https://foursquare.com/

> *Here, the physical location consists of the instant location of an individual at a given timestamp and the location history that an individual has accumulated in a certain period. Further, the interdependency includes not only that two persons co-occur in the same physical location or share similar location histories but also the knowledge, e.g., common interests, behaviors, and activities, inferred from an individual's location (history) and location-tagged data.*"

Zheng et al. sottolineano dunque come gli LBSN consistino in una nuova struttura sociale costituita da individui per i quali avviene una connessione tra la loro posizione fisica nel mondo reale e le risorse che essi condividono nel mondo virtuale.

1.3 Tecniche tradizionali di raccomandazione

Diversi sono gli approcci presenti in letteratura che possono essere impiegati nel processo di raccomandazione. In questo paragrafo sono presentate le tecniche tradizionali di filtraggio più diffuse: content-based, community-based e collaborative filtering. Ad esse si vanno a sommare tutti quegli algoritmi che combinano le caratteristiche di più approcci al loro interno e che sono pertanto catalogati come "ibridi".

1.3.1 Content-Based filtering

Un approccio molto comune nella progettazione di un Recommender System è il *Content-Based Filtering*. Esso si basa sulla descrizione dell'articolo da suggerire e sulle preferenze dell'utente, generando raccomandazioni derivate dalle scelte effettuate nel passato. Ad esempio, se l'utente ha ascoltato una determinata canzone, un sistema di raccomandazione in un contesto musicale suggerirà probabilmen-

te l'album di un artista affine alla canzone di partenza ma a lui inizialmente sconosciuto.

In un sistema di raccomandazione content-based i concetti sono descritti tramite parole chiave e un modello utente viene creato per indicare la categoria preferita degli oggetti apprezzati dall'utente, provvedendo così a consigli mirati. In particolare, vari item candidati sono comparati con quelli a cui precedentemente l'utente ha assegnato un voto; si stila dunque una classifica e sono raccomandati solo gli elementi più simili. Questo approccio affonda le sue radici nel campo di ricerca dell'Information Retrieval e dell'Information Filtering.

Per modellare le caratteristiche degli item nel sistema vengono applicate opportune funzioni. Una ampiamente usata è il *tf-idf*, soprattutto all'interno della rappresentazione noto come *Vector Space Model*. Per l'operazione di modellazione, il sistema si concentra prevalentemente su due tipi di informazione: le preferenze e lo storico delle interazioni dell'utente con il recommender system. Sostanzialmente questi metodi usano un profilo degli item per caratterizzarli all'interno del sistema, ad esempio un insieme di attributi e dimensioni discrete. Il raccomandatore crea un profilo content-based basandosi su un vettore pesato delle feature dell'item. Tali pesi denotano l'importanza di ciascuna dimensione per l'utente e possono essere calcolati con varie tecniche allo Stato dell'Arte, tra le quali machine learning, classificatori bayesiani e rete neurali.

Inoltre il sistema si può avvalere dell'analisi di feedback diretti di un utente, presenti solitamente nella forma di bottone di apprezzamento o non apprezzamento, nell'assegnazione di punteggi maggiori o minori.

Un problema chiave che si presenta nel filtraggio content-based è determinare se il sistema sia in grado o meno di apprendere i gusti degli individui a partire dalle loro azioni che riguardano una sorgente di dati e usarli anche in altri contesti. Ad esempio, un raccomandatore che conosce perfettamente le preferenze culinarie di una persona potrebbe avere difficoltà nel suggerire un album musicale che a questi potrebbe interessare.

1.3.2 Community-Based Filtering

Se in un filtraggio content-based ciascun utente viene considerato come uno scenario a sè stante, indipendente l'uno dall'altro, in un *Community- Based Filtering* una reciproca correlazione e le interazioni tra di essi risultano di fondamentale importanza.

Una frase evocativa per sintetizzare i concetti cardine su cui ruota questo approccio è "Dimmi chi sono i tuoi amici, io ti dirò chi sei". La vita quotidiana testimonia come le persone tendano a fidarsi maggiormente dei consigli degli amici rispetto a quelli provenienti da sconosciuti [SSS01].

Il fulcro di questo approccio è il concetto di *comunità*. Nel mondo scientifico sono conosciute anche con il nome di *cluster* o *moduli*. Con esse si intende, all'interno di una struttura a grafo, un sottoinsieme di vertici che possiedono proprietà comuni o giocano ruoli simili all'interno della rete [For10].

La società moderna offre numerose possibilità di formazione di gruppi, di respiro più o meno ampio: dalle famiglie, team di lavoro, cerchie di amici ad interi villaggi, città, nazioni. La diffusione di Internet ha anche contribuito alla creazione di insiemi di persone con una propria autonomia all'interno del mondo web, come le comunità virtuali.

Marco Tullio Cicerone stesso, in un famoso trattato, afferma che *" Nati sumus ad congregationem hominus et ad societatem comunitatem que generis humani."*, cioè che siamo nati con l'istinto dell'unione, dell'associazione e della comunanza propri del genere umano.

I social network stessi sono esempi pragmatici di grafi con comunità. Il concetto di omofilia precedentemente trattato assume quindi una valenza fondamentale in questo scenario.

Le comunità occorrono pertanto in una pletora di domini rappresentabili con una topologia a rete come la biologia, l' informatica, l'economia, la politica per citarne alcune e possiedono difatto applicazioni concrete. Si consideri ad esempio la loro introduzione nei sistemi di raccomandazione, motivo per cui i ricercato-

ri hanno introdotto in letteratura il community-based filtering. In tale ambito l'identificazione di cluster di clienti con interessi simili in una rete di relazioni d'acquisto tra i compratori e i prodotti di rivenditori online, consente di guidare i clienti nella lista degli articoli e migliorare pertanto le opportunità di business.

I Community-Based Recommender modellano e acquisiscono quindi le informazioni correlate alle interazioni sociali degli utenti e alle preferenze dei loro amici.

L'analisi delle comunità, estendendo lo spazio degli item, permette di ottenere benefici anche in relazione al problema della sparsità degli elementi e del cold-start, ovverosia dell'incapacità del sistema di inferire gli item o gli interessi dell'utente per i quali non ha ancora collezionato sufficienti informazioni.

1.3.3 Collaborative Filtering

Il *Collaborative Filtering* è una delle tecniche di raccomandazione che ha riscosso maggior successo e popolarità sia nel mondo scientifico che in quello commerciale. Analogamente al Community-Based Filtering, le raccomandazioni avvengono analizzando le preferenze di un gruppo di utenti, senza esaminare lo storico delle scelte e gli interessi specifici del solo utilizzatore. L'operazione di clusterizzazione è più specifica dell'altro approccio: infatti, per fornire i suggerimenti degli item a una persona, non si considera la comunità nel suo intero ma si vanno a selezionare, tra tutti, gli utenti con caratteristiche simili. In un'ottica collaborativa, si forniscono raccomandazioni o predicono i gusti sconosciuti degli altri utenti a partire dalle preferenze conosciute di un gruppo di persone. Infatti l'assunzione principale su cui si fonda questo approccio è che se una persona A ha la stessa opinione di una persona B su una determinata questione, allora è più probabile che, relativamente a un differente problema, A possieda la stessa idea di B rispetto a un altro individuo sconosciuto. Alcuni prerequisiti importanti per gli algoritmi di collaborative filtering sono: la partecipazione attiva dell'utente, un

modo semplice per rappresentare le preferenze e la capacità di abbinare utenti con interessi simili.

Un caso d'uso tradizionale prevede innanzitutto che l'utente esprima il proprio parere su di un determinato set di item assegnando a ciascuno di essi un voto. Con una certa approssimazione, ciò può essere considerato come una rappresentazione dei gusti dell'utente in quel determinato dominio. Il sistema effettua così il matching tra il ranking dell'utente corrente e quello degli altri utenti, cercando di massimizzare la percentuale di similarità. L'algoritmo di raccomandazione diventa così in grado di suggerire prodotti che sono stati maggiormente apprezzati da utenti con gusti simili ma non ancora visionati dall'utilizzatore del sistema.

I modelli proposti in letteratura per rappresentare gli item in un raccomandatore collaborativo sono vari. Fra questi figura il Vector Space Model, impiegato spesso anche nel Content-Based Filtering, mentre uno alternativo prevede una struttura a grafo.

Capitolo 2

Linked Open Data

I dataset Linked Open Data nascono dall'interpretare il Web come un'enorme repository di dati interconnessi tra di loro tramite *Universal Resource Identifier*(URI) e *Resource Description Framework* (RDF). Quest'ultimo è uno standard per descrivere e rappresentare dati e metadati nel Web e si si può rappresentare sotto forma di una struttura a grafo etichettato e orientato. I collegamenti RDF permettono di navigare da una risorsa web appartenente a una sorgente dati ad un'altra usando semplicemente dei browser web semantici. Inoltre questi link possono anche essere processati da crawler dei motori di ricerca del Semantic Web, i quali possono fornire sofisticate funzioni di query e ricerca. In aggiunta, siccome i risultati di tali interrogazioni sono dati strutturati e non solo link a pagine HTML, possono essere elaborati anche da altre applicazioni.

2.1 Annotazioni semantiche

L'annotazione, o *tag*, nell'ambiente informatico indica un termine o una parola chiave assegnata a una risorsa digitale, sia essa un'immagine digitale, un video o un semplice file, per aiutare nella descrizione e catalogazione dell'oggetto, permettendo che sia recuperabile tramite delle ricerche specifiche o una navigazione web. Un utente può etichettare liberamente e in modo assolutamenente informale

e personale i propri contenuti, usando il linguaggio naturale. In questo paragrafo si vede come i tag, che hanno notevolmente contribuito allo sviluppo e al successo del Web 2.0, rivestano un ruolo primario anche nel Semantic Web.

2.1.1 Il Semantic Web

> "*The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*"

Furono queste le parole con cui in un famoso articolo apparso sulla rivista *Scientific America* [TBLL01] nel 2001 Tim Berners-Lee, insieme a James Hendler e Ora Lassila, descriveva il Semantic Web. Da questa definizione emergono tre punti chiave: il Semantic Web, *estendendo* il Web attuale, si avvale dell'infrastruttura già esistente e usata quotidianamente; il suo obiettivo è fornire conoscenza alle risorse online, attribuendo un *significato ben preciso* all'informazione; in questo modo, favorendo la *cooperazione* tra persone e computer, le macchine possono fornire un valido supporto agli esseri umani nell'esecuzione e automazione dei compiti.

Per poter sviluppare queste potenzialità l'elaborazione dei dati deve essere attuata con l'impiego della logica: i computer devono poter accedere a collezioni strutturate di risorse (i *Linked Data*) ed essere in grado di ragionare automaticamente tramite insiemi di regole di inferenza. La sfida consiste nel trovare un linguaggio logico la cui espressività sappia nel contempo descrivere metadati, regole per la deduzione automatica, proprietà complesse ed evitare di essere talmente potente da introdurre anche paradossi. Due importanti elementi per lo sviluppo del Semantic Web sono il linguaggio di marcatura *XML*, acronimo di eXtensible Markup Language, e il modello *RDF*, acronimo di Resource Description Framework. XML permette ad ognuno di creare i propri tag, delle proprie etichette con cui annotare intere pagine Web o porzioni di esse; gli utenti possono

aggiungere strutture arbitrarie ai propri documenti ma senza fornire indicazioni sul significato di tale architettura. Non è quindi sufficiente per soddisfare la necessità dei sistemi di raggiungere un'*interoperabilità semantica*, quella capacità di interscambio di dati fra sistemi basata sul significato dell'informazione scambiata anzichè sul suo formato e sulla sua sintassi.

Un primo passo per ottenere questo livello di interoperabilità viene fornito dallo standard RDF, il modello relazionale dei dati definito e promosso dal W3C per l'interscambio di dati e metadati sul Web. I file RDF incapsulano la semantica in insiemi di *triple*, ognuna delle quali ricalca la struttura *soggetto - predicato - complemento oggetto* dei periodi semplici e viene codificata tramite tag XML. Le triple permettono di poter esprimere delle asserzioni sulle risorse, specificando quali entità (soggetto) abbiano proprietà (predicato) con certi valori (oggetto). Questa struttura diventa una modo naturale per descrivere la maggior parte dei dati comprensibili da una macchina, definendo termini o concetti che costituiscono lo schema logico dell'informazione da elaborare. Tutto ciò è raggiungibile in perfetto accordo con i principi e l'architettura del Web.

2.1.2 Ontologie per descrivere le annotazioni semantiche

Un'*annotazione semantica* consente di associare un significato ad una risorsa web in maniera formale. Una macchina diventa così in grado di capirne la semantica ed elaborarla di conseguenza. Il modo in cui vengono assegnate tali identificatori deve essere univoco ed evitare ambiguità: due sistemi distinti, infatti, potrebbero stabilire etichette differenti per esprimere la stessa risorsa e, pertanto, le asserzioni valide in un sistema non sarebbero confrontabili o integrabili con le asserzioni sulla stessa risorsa nel secondo sistema.

La soluzione viene offerta dalle *ontologie*. Nel mondo informatico esse sono documenti per la definizione rigorosa e formale delle relazioni tra gli oggetti, espresse attraverso un linguaggio di rappresentazione della conoscenza (RDF, OWL, DAMl+OIL...). Il termine proviene dal linguaggio filosofico, dove indica

quella branca della metafisica che si occupa delle teorie legate alla natura dell'essere in quanto tale. Un punto chiave dell'ambiente ontologico è l'osservazione che il mondo è composto da oggetti specifici che possono essere raggruppati in classi astratte in base a caratteristiche comuni. Aristotele, con il suo tentativo di catalogazione delle entità del mondo, ne è il maggiore esponenente filosofico.

In Informatica, ed in particolare nei sistemi di condivisione della conoscenza, gli oggetti del dominio che costituiscono l'universo del discorso sono a tutti gli effetti "le cose che esistono" e quindi devono essere rappresentate tramite ontologie. La definizione più completa e diffusa della sua applicazione al mondo informatico è fornita da Rudi Studer, Dieter Fensel e Richard Benjamins [SF98]:

> "*Ontologies are a formal, explicit specification of a shared conceptualization.*"

Questa citazione, seppur breve, racchiude tutto lo spirito dell'ontologia: *concettualizzazione*, quindi un modello astratto; *convidisa*, deve essere pubblica e utilizzabile da parte della comunità; *esplicita*, i concetti, le proprietà, le funzioni, le relazioni devono essere definiti esplicitamente; *formale*, cioè espressa in un linguaggio comprensibile dalle macchine e non in quello naturale.

Le tipologie di ontologie più diffuse sono la tassonomia e l'insieme di regole di inferenza. In particolare la *tassonomia* definisce le classi, sottoclassi degli oggetti e le relazioni che intercorrono tra essi, diventando un tool molto potente per gli utenti del Web. In questo modo è possibile esprimere un grande numero di associazioni tra le entità, assegnando proprietà particolari alle classi, e le sottoclassi possono ereditare tali proprietà. Un esempio ci viene fornito direttamente da Tim Bernes-Lee in [TBLL01]:

> " *The taxonomy defines classes of objects and relations among them. For example, an address may be defined as a type of location, and city codes may be defined to apply only to locations, and so on. Classes, subclasses and relations among entities are a very powerful tool for*

Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. If city codes must be of type city and cities generally have Web sites, we can discuss the Web site associated with a city code even if no database links a city code directly to a Web site."

2.2 RDF e grafi

La sigla RDF è l'acronimo di *Resource Description Framework* ed è lo standard W3C per rappresentare dati e metadati nel Semantic Web. Incarna pienamente il concetto delle annotazioni semantiche. Tutto ciò che viene rappresentato da tali espressioni è indentificato in una risorsa, sia essa un'intera pagina web, uno specifico documento XML o collezioni di pagine. Le risorse sono descritte tramite proprietà, ognuna con un significato ben definito che determina i valori ammissibili, i tipi di oggetti a cui fa riferimento e le relazioni con le altre proprietà.

La struttura base per descrivere un'entità RDF è l'*enunciato* o *statement* in inglese, una tripla soggetto - predicato - oggetto. Il *soggetto* è l'identificativo di una risorsa; il *predicato* indica la proprietà o l'attributo del soggetto che si vuole rappresentare; l'*oggetto* è il "valore" che il predicato assume in riferimento al soggetto. Per poter rendere le descrizioni delle risorse elaborabili da parte delle macchine, i componenti di un enunciato non sono semplici stringhe ma vengono indentificate univocamente attraverso un *Uniform Resource Identifier* (URL). Ad esempio l'enunciato, "`Ora Lassila è l'autore della pagina Web http://www.w3.org/Home/Lassila`" è caratterizzato da un soggetto (la pagina Web, indentificabile univocamente tramite la sua URL), un predicato ("avere un autore", per il quale si può ricorrere alla definizione di *Creator* dell'ontologia del Dublin Core) e un oggetto (Ora Lassila, l'autore della pagina). Come mostra la Figura 2.1, una singola tripla può essere disegnata con due nodi (soggetto e oggetto) e un arco diretto dal soggetto all'oggetto (predicato); piú triple concate-

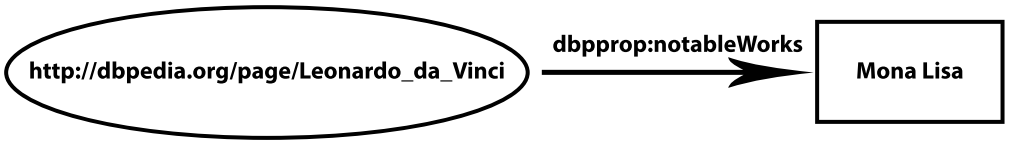

Figura 2.1: Rappresentazione grafica di una tripla RDF. Essa è formata da un soggetto, il cui URI è *http://dbpedia.org/page/Leonardo_da_Vinci*, un predicato il cui URI è *dbpprop:notableWorks* e un oggetto, *Mona Lisa*, il cui tipo è invece un letterale.

nate tra loro formano un grafo diretto etichettato. E' proprio in questa struttura dati che risiede la potenza espressiva del modello RDF, rendendolo un tassello fondamentale per il conseguimento del Linked Web of Data.

2.2.1 SPARQL

SPARQL (acronimo ricorsivo di SPARQL Protocolol and RDF Query Language) è un linguaggio per l'interrogazione di dati semantici. Esso basa la formulazione della query sul concetto di tripla e grafo, analogamente a quanto nel modello relazionale uno statement SQL fonda i propri componenti sulla nozione di ennuple e relazioni. Permette di interrogare dataset RDF in modo molto sofisticato, basandosi su algoritmi avanzati di *graph matching*. L'interrogazione è un pattern di un grafo RDF, la cui risposta è il sottografo che istanzia quel pattern, caratterizzato da un insieme di triple. La sintassi di SPARQL è molto ricca e prevede diverse famiglie di operazioni, come elencato in seguito:

SELECT query: è usata per estrarre i dati da un endpoint. I risultati sono restituiti generalmente in forma tabellare.

CONSTRUCT query: restituisce un grafo RDF che istanzia un graph pattern. Utile, ad esempio, per trasformare un vocabolario in un altro.

ASK query: ritorna un risultato booleano (true/false) per una query su un endpoint SPARQL, in base alla presenza o meno dell'informazione richiesta sulla base di conoscenza interrogata.

DESCRIBE query: è usata per estrarre, nella forma di grafo RDF, tutte le informazioni che l'endpoint SPARQL che si sta interrogando possiede sulle risorse indicate. Ad ogni server viene concessa la libertà di interpretare la query DESCRIBE in maniera diversa, rendendo così ciascuna operazione non interoperabile con le altre.

Una formulazione di query SPARQL è composta da tre sezioni principali: nella prima vengono dichiarati i prefissi utilizzati, relativi ciascuno a un ben determinato dataset; nella seconda si indica la tipologia di query da eseguire (SELECT, CONSTRUCT, ASK o DESCRIBE); nella terza, infine, si definisce il graph pattern che deve essere soddisfatto dalle triple che partecipano alla formazione del risultato. In quest'ultima sezione si sfrutta la clausola *WHERE* per tale operazione di filtraggio.

Un lettore interessato ad approfondire le specifiche del linguaggio può consultare il documento proposto dal W3C, "SPARQL Query Language for RDF"[1].

2.3 DBpedia

DBpedia è un attore molto importante nello scenario del Linked Open Data, tanto da venire raffigurato al centro della nuvola che rappresenta le varie interconnessioni semantiche tra i vari progetti che hanno aderito al Semantic Web, come testimonia la Figura 2.2. DBpedia, dal 2007, si pone il compito di estrarre le informazioni strutturate di Wikipedia[2] e rilasciarle successivamente sul Web come Linked Open Data in formato RDF. La base di conoscenza di DBpedia, al momento della stesura di questa tesi, descrive 4 milioni di concetti, di cui 3.22

[1] http://www.w3.org/TR/rdf-sparql-query/
[2] www.wikipedia.org

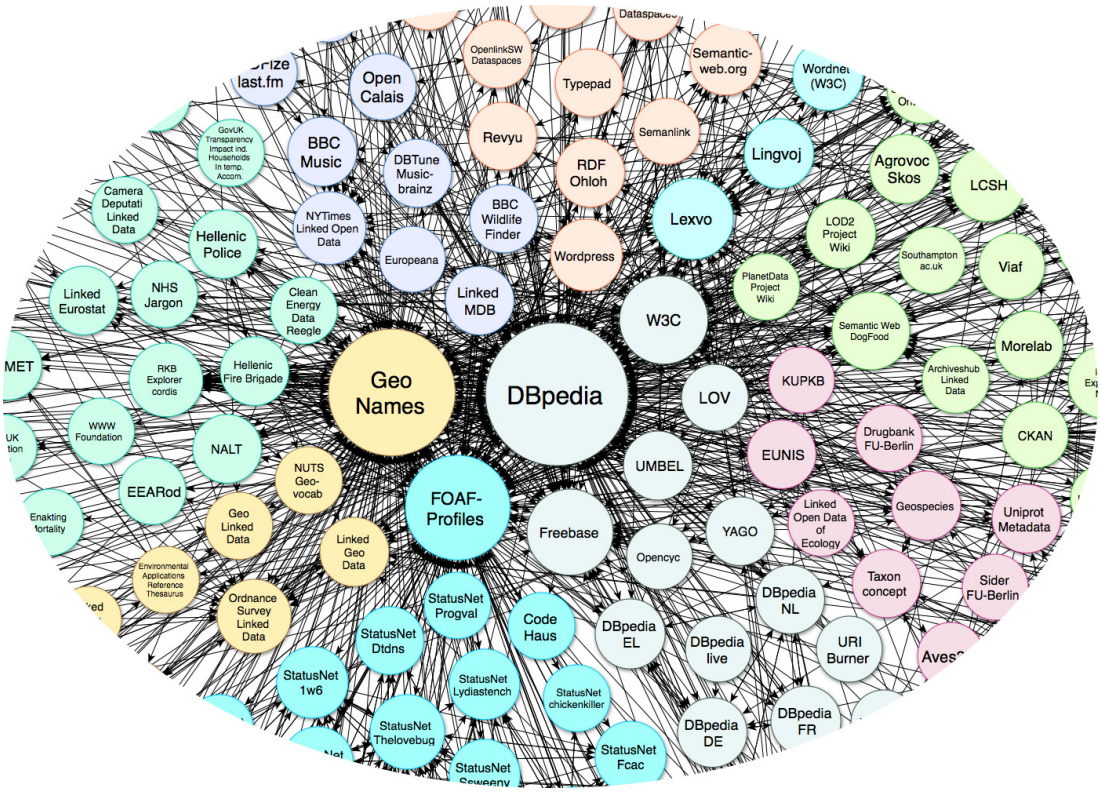

Figura 2.2: Porzione centrale del Linked Open Data Cloud aggiornata ad Agosto 2014. Essa mostra le interconnessioni semantiche che sussisistono tra i vari progetti del Semantic Web. Da notare come DBpedia sia collocata esattamente al centro di tale rete. Anche Europena possiede una posizione molto centrale. Da sottolineare anche la divisione in colori, in base al dominio a cui ciascun progetto appartiene (Scienze naturali, Pubblicazioni, Geografiche, Governi...). Per avere uno sguardo globale della LOD Cloud, si rimanda il lettore alla figura originale, pubblicata sul sito `http://lod-cloud.net`, i cui autori sono Max Schmachtenberg, Christian Bizer, Anja Jentzsch e Richard Cyganiak.

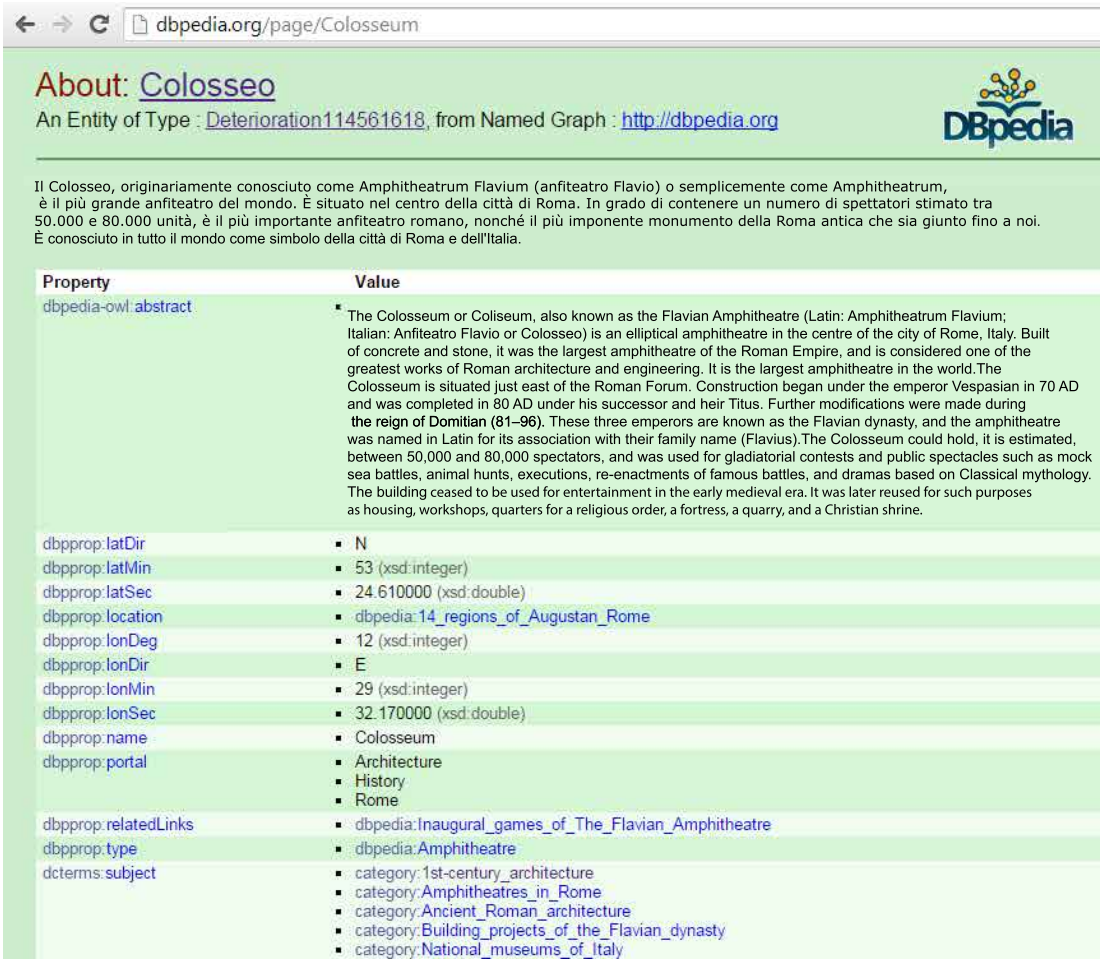

Figura 2.3: Pagina di DBpedia della risorsa relativa al Colosseo.

milioni sono classificati secondo un'ontologia consistente, comprendendo 832,000 persone, 639,000 posti, 372,000 lavori creativi (tra cui 116,000 album musicali, 78,000 film e 18,500 videogiochi), 226,000 specie animali e vegetali, 209,000 organizzazioni (49,000 compagnie e 45,000 istituti di istruzione), e 5,600 malattie. La Figura 2.3 illustra un esempio di risorsa fornita da DBpedia.

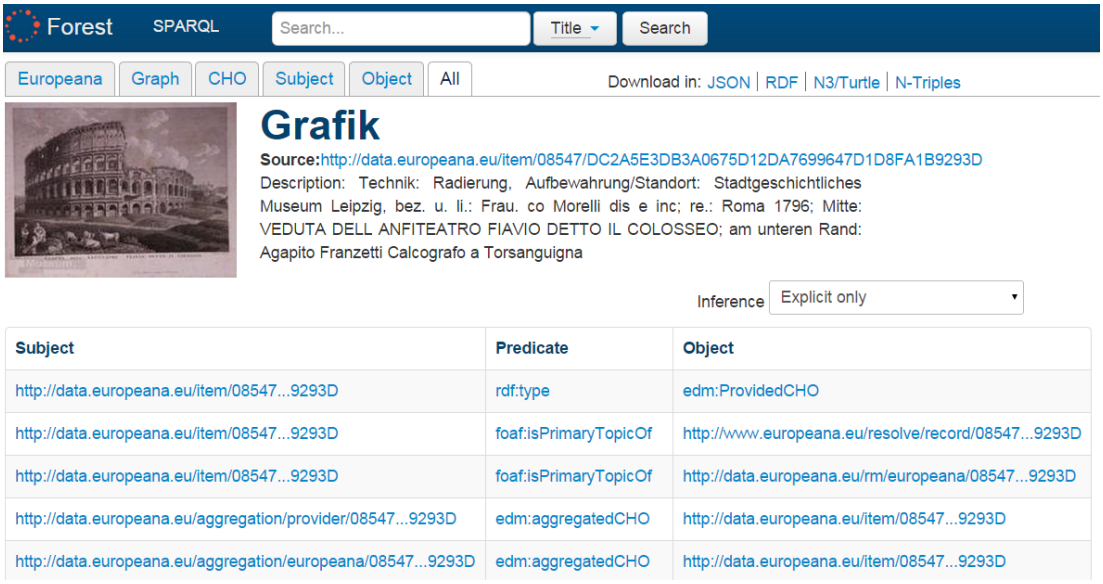

Figura 2.4: Pagina di una risorsa Europeana che ha il Colosseo come soggetto principale. In questo caso si tratta di una foto del monumento romano proveniente dal museo tedesco “Stadtgeschichtliches Museum Leipzig”.

2.4 EUROPEANA

Europeana è una biblioteca digitale europea inaugurata il 28 Novembre 2008. E’ accessibile tramite un portale web ed espone agli utenti milioni di libri, dipinti, film, musiche, opere d’arte, rappresentazioni virtuali di luoghi o architetture in 3D, archivi, che sono stati pubblicati in formato digitale in Europa. La Gioconda di Leonardo Da Vinci, la Ragazza con l’orecchino di perla di Johannes Vermer, le foto della Prima Guerra Mondiale, i lavori di Charles Darwin, le musiche di Ludwig van Beethoven sono solo alcuni degli esempi di opere che sono consultabili su Europeana. Questo progetto ha raccolto contribuiti da oltre 2000 istituzioni europee. Tra di essi figurano non solo i principali nomi internazionali come Rijksmuseum di Amsterdam, il British Museum di Londra o il Louvre di Parigi, ma anche archivi regionali e musei locali dei 28 Paesi membri dell’Unione Europea. Insieme sono state assemblate numerose collezioni per consentire agli utenti di esplorare l’immenso patrimonio artistico e culturale europeo, dalla preistoria fino

al periodo attuale.

Le opere esposte in Europeana sono di proprietà delle rispettive istituzioni culturali e sono loro le responsabili dell'hosting. Il database di Europeana contiene, invece, tutti i metadati relativi a tale opere, tra cui titolo, creatore, descrizione, museo proprietario, soggetto, diritti, provider, paese di provenienza, formato del file, tipo. Tali dati sono liberamente accessibili sotto forma di Linked Open Data, interrogandone l'endpoint SPARQL, o semplicemente tramite il browser web fornito dal portale. L'ontologia per rappresentare i contenuti della biblioteca è *Europeana Data Model* (EDM). Essa agisce come una comune ontologia ad alto livello che conserva i modelli dei dati originali e le prospettive dell'informazione, attribuendo contemporaneamente un'alta priorità alla interoperabilità semantica con le ontologie principali, come *SKOS*[3], *FOAF*[4] o *Dublin Core*[5]. Prevede, inoltre, un duplice approccio di rappresentazione semantica: uno è orientato agli oggetti, l'altro agli eventi. Una classe ontologica centrale in EDM è *edm:Agent*: essa comprende persone o agenti, considerati sia singolarmente che in gruppi. Esempi sono Leonardo Da Vinci o il British Museum. *edm:EuropeanaObject* rappresenta invece un qualsiasi oggetto che sia prodotto dalle attività di Europeana, come un'annotazione creata da un utente durante una sua interazione con il portale. *edm:PhysicalThing* si usa invece per indicare un elemento fisico persistente, come un dipinto, un libro o un'opera d'arte in generale, ma non per le persone (già coperte da Agent).

2.5 NoSQL

Le tecnologie *NoSQL (Not only SQL)* introducono DBMS basati sul modello key-value che è una rappresentazione nativa di un grafo RDF. Si tratta di una nuova generazione di database management system non relazionali che ha preso piede

[3]http://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html
[4]http://xmlns.com/foaf/spec/
[5]http://dublincore.org/

negli ultimi anni per rompere con la lunga tradizione di basi di dati relazionali. Essi puntano ad essere distribuiti, open source, senza uno schema tabellare fisso, scalabili orizzontalmente. Le attuali soluzioni NoSQL si possono largamente suddividere in quattro macrocategorie: i *Key-value database* sono caratterizzati da tabelle di hash distribuite e Redis[6] ne è un esempio; i *document database*, come MongoDB[7], presentano sempre un approccio chiave-valore ma i valori sono memorizzati come dati semi-strutturati; i database *wide-coloumn* che ricalcano il modello di BigTable[8] di Google e presentano un sistema di storage distribuito, per elaborare dati strutturati, in grado di scalare efficacemente e supportare carichi al livello del petabyte di dati attraverso migliaia di server; *graph database* che memorizzano i dati in un grafo, una struttura dati flessibile in grado di rappresentare elegantemente un qualsiasi tipo di dati, in particolare quelli connessi.

Neo4j[9] presenta uno storage pienamente ottimizzato per salvare strutture a grafo, assicurando massime perfomance e scalabilità. È in grado di lavorare su dataset di miliardi di nodi, relazioni e proprietá su un singolo calcolatore e può scalare su multiple macchine. Usando un framework molto potente, Neo4j garantisce un'efficiente navigazione nel suo modello tramite algoritmi ottimizzati di ricerca in ampiezza o, in alternativa, sottoponendo query nel linguaggio dichiarativo *Cypher*, sviluppato dalla stessa compagnia. Neo4j espone driver in numerosi linguaggi di programmazione, Java in primis, e si presenta nella forma di server o come un'istanza embedded. La versione rilasciata al momento della stesura della tesi ed utilizzata in Cicero è la 2.1.5.

[6] http://redis.io/
[7] http://www.mongodb.org/
[8] http://research.google.com/archive/bigtable.html
[9] www.neo4j.org/

Capitolo 3

Stato dell'Arte

La letteratura dei sistemi di raccomandazione è molto ricca e copre domini applicativi molto vasti. Da un lato, infatti, è presente un cospicuo numero di articoli che propongono alla comunità scientifica tecniche di raccomandazione innovative, dai tradizionali approcci content-based e collaborative filtering a loro versioni più evolute. Dall'altro, sono stati introdotti sistemi allo Stato dell'Arte in cui tali proposte sono state effettivamente sviluppate e analizzate, studiandone gli aspetti positivi e negativi. Questo capitolo si concentra in particolar modo su quei sistemi di raccomandazione che sperimentano l'apporto fornito sia dal Social Web che dal Linked Open Data per la modellazione utente e l'esplorazione degli item da suggerire. Il bisogno di una rappresentazione semantica dei dati e dei profili utenti è considerato sempre di più una delle prossime, maggiori sfide nel campo dei sistemi di raccomandazione.

3.1 Espandere la modellazione utente sul Social Web con i Linked Open Data

Il lavoro presentato in [AHHT12] è probabilmente quello con cui Cicero presenta maggiori analogie, non solo dal punto di vista del dominio di raccomandazione in

cui questi si va a introdurre, ma anche da quello delle tecniche e degli algoritmi di cui si avvale. Gli autori di questo articolo illustrano un framework per arricchire la modellazione utente sul Social Web con le informazioni estratte dal Linked Open Data. Esso monitora le attività utente su piattaforme sociali, analizzando in particolare lo stream di Twitter e la condivisione di risorse su Flickr, inferisce il significato semantico delle attività utente e prevede strategie per collezionare dati di background dal Semantic Web, al fine di generare profili utenti robusti semanticamente che supportino una data applicazione.

La modellazione utente è *geospatial-centric*. Si basa, infatti, sulla strategia che, dato un insieme di punti di interesse e un utente u, assegna a ciascun POI p un peso che rifletta l'interesse che u mostri in p. Vengono considerati come dati utente le informazioni semantiche, in termini di URI DBpedia, connesse a POI (dbpedia:Place), estratte dai tweet del profilo Twitter dell'utente o dai metadati delle foto che questi ha condiviso su Flickr, estraendo, se possibile, le coordinate GPS in cui esse sono state scattate. Come strumento per il riconoscimento di tali POI, il framework descritto si avvale dei servizi di *Named Entity Recognition* forniti da Geonames[1].

I valori di background relativi a tali concetti geospaziali estratti ed espressi nella forma di statement RDF, sono ottenuti dai dataset di DBpedia. Per associare le risorse utente con i POI, gli autori dell'articolo usano in particolare tre pattern sui grafi: *menzione diretta*, *menzione indiretta di primo livello* e *menzione indiretta di secondo livello*. Il primo si riferisce al caso in cui il concetto di interesse sia esplicitamente presente nei dati utente; il secondo alla situazione in cui il concetto menzionato appartenga alla stessa tripla RDF del concetto di interesse; il terzo pattern, invece, si verifica quando, all'interno del grafo RDF, è necessaria una navigazione a profondità due per raggiungere il concetto di interesse a partire dal concetto citato. Per assegnare un punteggio di preferenza ad un punto di interesse, attribuiscono un peso basato sul numero di occorrenze. A

[1] www.geonames.org/

tal proposito contano il numero di attività utente, rappresentate come concetti semantici estratti, che presentino un match con un pattern usato dalla strategia di modellazione utente. In totale ottengono nove strategie differenti da analizzare: menzione diretta, indiretta primo livello, indiretta secondo livello da applicare al caso in cui sia disponibile solo lo stream delle attività su Twitter, solo le risorse condivise su Flickr e quello in cui l'utente sia attivo su entrambi i social network.

La loro fase di sperimentazione è stata portata avanti con metodologie system-oriented, soffermandosi sulle metriche di precision, F-measure e recall per saggiare la bontà delle raccomandazioni. In particolare ha sfruttato un vasto dataset composto da più di 2.4 milioni di tweet e 800.000 foto di Flickr, raccolte monitorando l'attività di 394 utenti nell'arco temporale di un anno. Il risultato della valutazione ha rivelato che l'aggregazione dei dati sociali da entrambe le sorgenti favorisce l'apprendimento da parte del sistema delle preferenze utente. Inoltre, ha evidenziato come l'impiego delle informazioni contestuali derivate dal Linked Open Data conducano a miglioramenti sostanziali nell'efficacia della modellazione utente.

3.1.1 Confronto con Cicero

Le metodologie e gli algoritmi implementati in Cicero saranno ampiamente sviluppati nei prossimi capitoli. Tuttavia si vogliono ora mostrare brevemente i punti in comune che presenta con il sistema proposto dal gruppo di ricerca dell'Università di Delft. Anche in Cicero si crea un profilo iniziale dell'utente, analizzando le attività di account su un Location-Based Social Network ed espandendolo con il contesto semantico fornito dal Linked Open Data. Al posto di Twitter e Flickr si è preferito usare Facebook, vista la sua capillare diffusione, senza però perdere in funzionalità: questi infatti presenta sia la possibilità di analizzare le foto, sia gli status dell'utente, per carpire le entità semantiche. Per effettuare le operazioni di Named Entity Recognition si combinano i servizi di GeoNames e TagMe[2] e la loro accuratezza è migliorata tramite opportuni algoritmi di disambiguazione

[2]http://tagme.di.unipi.it/

sviluppati in Cicero. Come basi di conoscenza del Semantic Web viene adoperata non solo DBpedia, ma anche Europeana. Inoltre, dal punto di vista degli algoritmi di raccomandazione, Cicero, al posto di una strategia basata sul numero di occorrenze di match con un graph pattern, implementa tecniche di collaborative e community-based filtering e ricerca di luoghi che siano semanticamente e geograficamente vicini agli item suggeriti all'utente. Infine la sperimentazione in Cicero presenta un approccio user-centric ed è condotta tramite opportuni questionari sottosposti agli utenti valutatori.

3.2 Cinemappy

Cinemappy [ONM+12] è un'applicazione mobile location-based per la raccomandazione contestuale di film. Essa affina i risultati di un sistema di raccomandazione content-based, sfruttando da un lato le informazioni contestuali collegate alla posizione corrente dell'utente, sia in termini spaziali che temporali, dall'altro, la principale sorgente di dati LOD, ovverosia DBpedia.

In particolare, il sistema consiglia i film da guardare nei cinema ubicati in prossimità dell'utente. Dalla base di conoscenza di DBpedia vengono estratte le caratteristiche salienti del film, come i generi, gli attori, i direttori, i registi e via discorrendo. I contributi principali apportati dagli autori dell'articolo alla letteratura sono sintetizzabili nei seguenti:

- la proposta di un recommender system basato sulla semantica e sul contesto;
- l'integrazione di sorgenti eterogenee di informazioni per la raccomandazione di video, in particolare dati provenienti dal Web tradizionale e dai Linked Open Data;
- la dimostrazione che le applicazioni Android che si avvalgono sia della geolocalizzazione che dell'analisi del contesto, possono essere sviluppate effi-

cacemente per raccomandare film in onda nei cinema ubicati nei dintorni dell'utente.

L'assunzione basilare su cui si fonda l'approccio di Cinemappy è che due film sono collegati tra loro se possiedono caratteristiche comuni, come attori, genere, pubblico, regista. Maggiore è il numero di punti condivisi, maggiore è la loro somiglianza. La similarità tra due risorse può essere quindi riconosciuta in vari modi a partire da un grafo RDF. Ad esempio quando esse appartengono allo stesso statement e, di conseguenza, sono direttamente correlate; in alternativa quando sono soggetti di due triple che hanno la stessa proprietà e lo stesso oggetto oppure se sono oggetti di due triple con cui condividono oggetto e predicato.

Per calcolare il grado di similarità tra due film, in Cinemappy è stato adottato il Vector Space Model (VSM), uno dei modelli classici più popolari nel contesto dell'Information Retrieval. In questo modo il sistema è in grado di rappresentare gli item come vettori di features e computare la misura di similarità tra essi con tecniche allo Stato dell'Arte.

3.3 Quickstep e Foxtrot

Alcuni sistemi di raccomandazione basati su ontologie sono presentati in [MRS09]. In questo paper gli autori descrivono un approccio che sfrutta le ontologie per fornire raccomandazioni. Nell'articolo sono introdotti *Quickstep* e *Foxtrot*, due sistemi sperimentali che effettuano collaborative filtering sugli item attraverso una modellazione utente fondata sulla semantica. Tali profili sono rappresentati da argomenti riguardanti paper accademici di ricerca, relativi a una data ontologia. Le raccomandazioni sono calcolate facendo corrispondere i topic dell'utente corrente con quelli di profili utente con interessi simili. L'approccio principale è, quindi, quello di un filtraggio collaborativo all'interno dello spazio degli item.

Gli autori dell'articolo dimostrano varie tesi:

- l'inferenza ontologica migliora la profilazione utente;
- la conoscenza derivata dalle ontologie facilita l'avviamento iniziale del sistema, contrastando il problema del cold-start che affligge solitamente i sistemi di raccomandazione;
- visualizzare i profili migliora l'accuratezza della modellazione.

Queste conclusioni giungono a valle di un'analisi condotta su vari casi di studio: inizialmente due esperimenti di dimensioni ridotte, comprendenti un dataset di 24 soggetti monitorati nell'arco di 3 mesi e, successivamente, uno di scala decisamente più ampia, composto da 260 soggetti durante un intero anno accademico.

3.4 Modelli ibridi di raccomandazione basati sulle ontologie

Gli autori del paper [CBC08] propongono un sistema di raccomandazione ibrido in cui le preferenze utente e le feature degli item sono descritte in termini di concetti semantici, definiti all'interno di ontologie di dominio. Aspetti chiave del loro lavoro consistono nello sfruttamento dei metadati degli oggetti raccomandati e dei profili utente in un modo generale e portabile, insieme alla capacità di inferire la conoscenza dalle relazioni presenti nelle ontologie. In poche parole, i concetti, gli oggetti e gli spazi utente sono clusterizzati in modo coordinato. Gli insiemi così creati vengono usati per cercare la similarità tra gli individui su più livelli semantici. Tali strati corrispondono a implicite Comunità di Interesse, in cui sono connessi utenti con interessi simili. Esse consentono raccomandazioni collaborative con precisione potenziata. In particolare, gli spazi ontologici assumono la forma di una rete semantica di concetti correlati tra loro. I profili utente, invece, sono descritti inizialmente come vettori pesati che misurano quanto quei concetti risultino importanti per l'utente.

Il fondamento logico su cui si basa la suddivisione in strati è che, seppur due individui convidono la passione per un argomento, ad esempio il cinema, non è detto che abbiano interessi comuni in un altro dominio, come lo sport. Le opinioni di queste persone sui film possono risultare di grande valore reciproco e rischiano, tuttavia, di essere ignorate dai classici algoritmi di raccomandazione collaborativi: infatti la similarità globale tra i due utenti potrebbe essere molto bassa. Da qui sorge l'idea di distinguere livelli differenti all'interno degli interessi e delle preferenze degli utenti, allo scopo di garantire suggerimenti più raffinati e sensibili al contesto.

L'approccio degli autori è stato da loro testato con due differenti serie di esperimenti: la prima include i profili definiti in modo manuale da utenti reali; la seconda, invece, coinvolge i modelli generati automaticamente a partire dai dati provenienti da *IMDb*[3] e *MovieLens*[4], entrambi importanti dataset nell'ambito dei Linked Open Data.

3.5 Dbrec e le raccomandazioni musicali basate su DBpedia

In [Pas10] viene proposto *dbrec*, un sistema di raccomandazione musicale che sfrutta i dataset di DBpedia e offre suggerimenti per oltre 39000 band e artisti solisti.

L'autore dell'articolo discute su come le misure di distanza semantica possano essere usate per identificare le relazioni tra risorse, applicate allo scenario dei Linked Open Data ed essere usate per fornire raccomandazioni in ambito musicale. A tal proposito si definisce il concetto di *Linked Data Semantic Distance (LDSD)*, ovvero distanza semantica nei Linked Data, con l'obiettivo di trovare la distanza semantica tra le risorse. Nello specifico, vengono introdotte tre tipologie di *LDSD*:

[3] www.imdb.com/
[4] https://movielens.org/

distanza diretta, che si basa sui link diretti tra le risorse, sia entranti che uscenti; *distanza indiretta*, in cui si considerano anche le relazioni indirette, in quanto il valore dei Linked Data risiede non solo nei link diretti tra le risorse, ma anche sulle connessioni condivise con altri elementi; *distanza combinata*, nella quale i link diretti sono combinati con quelli indiretti.

Obiettivo del lavoro è scoprire come la misura di distanza semantica possa essere applicata a risorse pubblicate sul Web come Linked Open Data, considerando in particolare le caratteristiche salienti del Semantic Web. In altri termini, dbrec si affida solamente ai link, senza preoccuparsi dei valori letterali e della loro vicinanza linguistica e sintattica, ai dati delle risorse, senza concentrarsi molto sullo sviluppo di ontologie descrittive, ed agli URI. In questo modo, infatti, le distanze possono essere calcolate semplicemente accedendo agli URI e recuperando le triple RDF corrispondenti.

La sperimentazione di dbrec segue un approccio user-centric che coinvolge 10 individui. Essa è volta a comparare questo sistema con altri prodotti commerciali consolidati, come Last.fm [5], analizzando il modo in cui in cui gli utenti assegnano un voto alle sue raccomandazioni.

3.6 Raccomandazione di foto con modellazione utente su social network.

Gli autori di [EKH13] introducono un approccio *cross-domain* per la modellazione utente personalizzata che acquisisce informazioni contestuali dai Linked Open Data ed inferisce le preferenze del profilo analizzando le sue attività sulla piattaforma di Facebook. Tali basi di conoscenza sono quindi analoghe a quelle adottate in Cicero. Siccome l'obiettivo degli autori dell'articolo è la raccomandazione di foto socialmente rilevanti, espandono il loro modello utente collegandolo con i servizi forniti da Flickr.

[5] http://last.fm

Nello specifico, essi collezionano i dati sociali riguardanti l'utente, come i suoi amici e le loro foto. Tali risorse possiedono vari metadati associati, come titolo, descrizione, posizione, data di upload, commenti, like, tag. A proposito dei tag, essi argomentano come vi sia una differenza sostanziale tra le annotazioni di Flickr e quelle sociali di Facebook. Infatti, mentre in Flickr i tag sono comparabili ad etichette con cui descrivere ed organizzare le foto, facilitandone una successiva ricerca, su Facebook sono associati solo a fotografie che rappresentano qualcosa o qualcuno. Sono un modo alternativo per comunicare che la persona identificata dal tag è stata ripresa in quella istantanea. Gli autori dell'articolo ragionano anche su un trend emergente per cui gli individui vengono taggati non tanto in quanto effettivamente presenti in tale immagine, ma perchè il suo possessore vuole che l'utente taggato veda la risorsa condivisa (infatti questi riceve una notifica non appena il tag gli viene attribuito). Ad ogni modo, il tag su Facebook suggerisce l'importanza della foto per le persone in essa etichettate. Gli autori assumono, di conseguenza, che l'importanza del tag in entrambi casi sia di fatto equivalente.

I dati sociali estratti da Facebook vengono convertiti in statement RDF e salvati così sul Triple Store di Virtuoso[6]. A tal proposito, prevedono un collegamento a risorse di DBpedia esistenti, effettuando un riconoscimento con DBpedia Spotlight[7], un sistema di Named Entity Recognition che mappa le stringhe di input in URI di pagine DBpedia. Non vi è, tuttavia, un'arricchimento della disambiguazione con le informazioni geolocalizzate della foto come avviene invece in Cicero (vedi capitolo 4), soluzione che conduce ad un miglioramento dell'accuratezza nei sistemi di NED.

Inoltre, come ontologia per la rappresentazione semantica delle caratteristiche degli utenti, gli autori dell'articolo hanno adottato *FOAF*, letteralmente *Friend Of A Friend*. Essa fornisce nativamente proprietà con cui descrivere il profilo della persona, come l'identità, la parentela, le amicizie, le sue attività. Esempi sono

[6]http://virtuoso.openlinksw.com/
[7]http://spotlight.dbpedia.org/

le proprietà *foaf:familyName* o *foaf:knows*, con cui si denota, rispettivamente, il cognome e l'amicizia con un altro individuo . L'ontologia base di FOAF è stata estesa con classi semantiche create da loro ad hoc, come *SocialThing*: essa descrive elementi che sono onnipresenti nei social network ma non coperti da FOAF.

FOAF primariamente mira a descrivere una persona in quanto essere umano, non le sue attività sui social network. Il sistema proposto dall'articolo usa quindi le interazioni dell'utente per calcolare quanto siano strettamente legati due amici, con l'assunzione che un numero maggiore di interazioni implichi relazioni più strette. Si contano così le occorrenze di un commento ad una determinata foto di un amico, il numero di like, i tag.

Al termine della modellazione delle informazioni estratte dagli account Facebook e Twitter in termini di triple RDF, il sistema è in grado di provvedere alla raccomandazione di foto personalizzate. Esse sono generate inferendo gli interessi dei migliori amici dell'utente a partire da un'analisi del loro profilo aggregato, composto dalla lista con la classifica degli amici *R(u)* e dagli interessi utente *I(u)*. La classifica *R(u)* include gli amici ritenuti più vicini e più stimati ed è compilata esaminando le loro mutue interazioni (come *like* e commenti sulle foto) e la loro generale reputazione nel social network. Investigano quindi la proprietà *foaf:depiction* per vedere se l'individuo compare in ulteriori immagini. Gli autori assumono, infatti, che se due utenti compaiono reciprocamente nelle foto dell'altro, è alta la probabilità che siano interessati nelle foto pubblicate l'un l'altro.

Per quanto riguarda *I(u)*, l'interesse dell'utente è descritto attraverso le pagine che questi ha apprezzato con un like su Facebook, includendo come metadati gli elementi marcati come interessi, libri letti, l'attuale geolocalizzazione dell'utente, il fatto o meno di raffigurare un amico dell'utente nella foto. Altre informazioni delle immagini che vengono salvate sono i loro metadati su Facebook e Flickr, come la didascalia o il nome dell'album, rappresentati con la proprietà *SocialTag*.

Al fine di raccomandare una fotografia per un utente, gli autori dell'articolo

prendono in considerazione gli interessi non solo dell'utente stesso, ma anche quelli dei suoi amici più simili. L'approccio alla raccomandazione è difatto collaborativo (analogamente a quanto avviene in Cicero) e fondato sul modello Vector Space, in cui le dimensioni spaziali sono tutti i possibili tag sociali e interessi.

Capitolo 4

Caratteristiche teoriche del sistema Cicero

Cicero, il framework qui proposto, è un sistema di raccomandazione sociale di beni artistici e culturali appartenti al patrimonio mondiale. Analizzando le attività di un account del social network Facebook, esso costruisce un profilo utente iniziale e ne arricchisce la modellazione con informazioni semantiche provenienti da basi di conoscenza dei Linked Open Data, in particolare da DBpedia ed Europeana. Questo capitolo si concentra principalmente sui fondamenti teorici su cui si basa il sistema Cicero.

4.1 Il sistema Cicero

Cicero (nome latino di Marco Tullio Cicerone) è stato uno dei personaggi più influenti ai tempi dell'antica repubblica romana, simbolo dell'eloquenza romana e autore di numerosi documenti letterari.

In onore dell'oratore latino, a partire dal XVIII secolo il nome di Cicero è stato attribuito per antonomasia alle guide che conducono i turisti in visita artistica a Roma. Analogamente, il sistema qui proposto vuole essere un punto di riferimento per tutti quegli utenti che desirano essere virtualmente accompagna-

ti alle scoperte del patrimonio artistico e culturale presente nel nostro pianeta, ricevendo suggerimenti personalizzati.

Molte volte capita di sfogliare numerose guide turistiche cartacee o perdersi nell'immensità delle pagine di Internet alla ricerca di consigli utili, soprattutto quando si vuole visitare una nuova città che non si conosce. Con un solo, semplice passaggio, l'applicazione Cicero viene incontro a tali esigenze, proponendo ai suoi utilizzatori opere d'arte e monumenti da visitare in modo automatico e veloce. Ciò migliora decisamente l'esperienza degli utenti appassionati di viaggi e d'arte, fornendo dei consigli tarati alle loro preferenze, inferite attraverso un'accurata analisi dei loro profili utente su Facebook. Si ottiene, pertanto, l'effetto positivo di ridurre drasticamente lo stress legato alla fase di selezione di cosa visitare.

Il processo di modellazione utente si snoda su tre momenti principali. Nella prima fase vengono collezionate le informazioni sociali. Il processo di estrazione è implicito e trasparente all'utente. Successivamente, tali dati sono resi persistenti nel graph database Neo4j. Infine vi è la fase di raccomandazione vera e propria.

Di tutta la comunità presente sul social network, il sistema esamina il cluster caratterizzato dall'utente e dai suoi amici, diverso quindi da uno sperimentatore all'altro.

4.2 Facebook

Facebook[1] è un social network nato nel 2004 che, nel corso di pochi anni, ha acquisito una popolarità virale. Leader del mercato, con oltre 1,3 miliardi di account registrati è attualmente il social network più diffuso in tutto il mondo, come testimonia la Figura 4.1.

E' stato scelto non solo per questo motivo, ma anche per la possibilità di avere post geolocalizzati grazie al servizio *Facebook Places*[2].

[1] https://www.facebook.com/
[2] https://www.facebook.com/places/

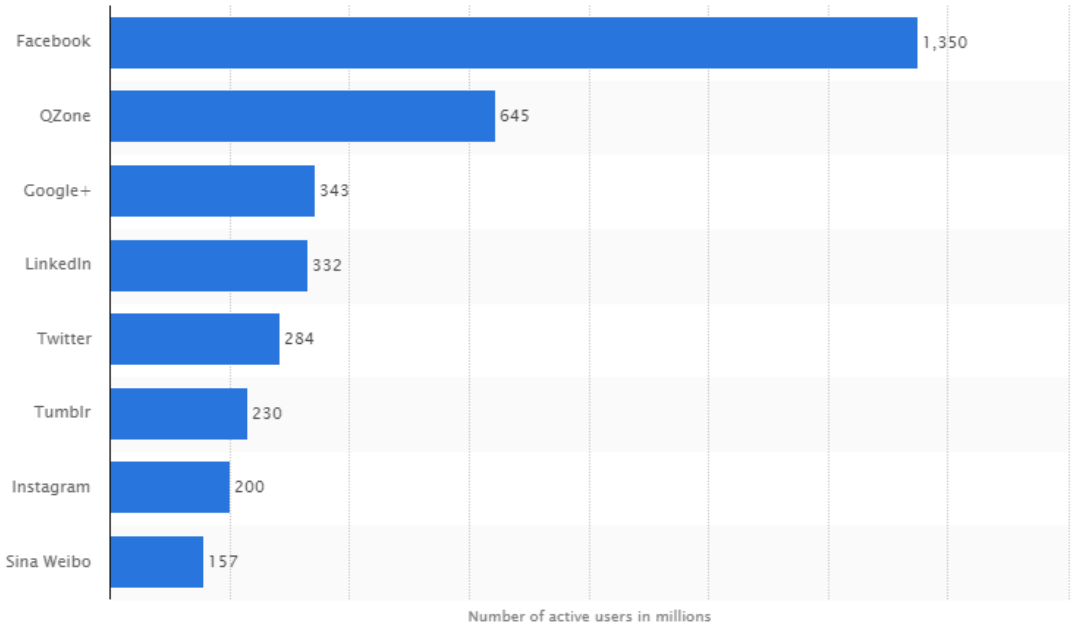

Figura 4.1: Statistiche aggiornate a Novembre 2014 relative al numero di account attivi sui social network più diffusi nel mondo. Il leader del mercato è Facebook, il primo social network ad aver superato il miliardo di account registrati. A seguire nella top five si collocano QZone (645 milioni), Google+ (343 milioni), LinkedIn (332 milioni) e Twitter(284 milioni). Questo istogramma proviene dal portale statistico `http://www.statista.com`.

Ogni persona iscritta a Facebook possiede un account con le informazioni di accesso. Ogni account può avere un diario personale e gestire più pagine.

I diari personali, che a volte chiamiamo anche profili, sono rivolti ad un uso individuale e non commerciale. Rappresentano singole persone e devono essere creati con i nomi individuali. I diari possono essere seguiti per vedere gli aggiornamenti pubblici delle persone a cui si è interessato ma con cui non si è amici. Una *Facebook Page*[3] è simile a un diario personale, ma offre strumenti esclusivi per connettere le persone a qualcosa dei cui aggiornamenti vogliono essere notificate, come un'azienda, un marchio, un'organizzazione o un personaggio famoso. Essa

[3]https://www.facebook.com/about/pages

è una pagina web che può rappresentare luoghi fisici, marchi, aziende, comunità, artisti musicali e illustra le loro caratteristiche salienti. Le Facebook Page sono gestite da persone che hanno un diario personale, non sono account Facebook indipendenti e vi si può accedere con le stesse credenziali dell'utente che le ha create. E' possibile cliccare sul bottone *Like* (Mi piace) di una Facebook Page per vedere gli aggiornamenti nella sezione Notizie ed essere così continuamente avvertiti con una notifica qualora si verifichino degli aggiornamenti di stato relativi ad essa.

4.3 Estrazione dei dati sociali da Facebook

Il sistema di raccomandazione Cicero prevede metodi per l'estrazione e la raccolta dei dati sociali a partire dallo stream delle attività degli account sulla piattaforma Facebook. La prima fase consiste nell'analisi degli endpoint *feed*, *post*, e *photo* della *Graph API*[4] fornita dal social network. L'obiettivo è quello di esaminare tutti gli status e le foto pubblicate sulla bacheca dell'utente alla ricerca di informazioni geolocalizzate allegate ad essi. Esse possono presentarsi nella forma di *tag* esplicito o essere estrapolate dal testo del messaggio. Si prenda in considerazione, ad esempio, il post "Quanto sei bella Roma! PRESSO *Colosseo*", estratto dalla bacheca di un utente, come mostra la Figura 4.2. Nel primo caso si salverà il tag esplicito Colosseo, con le relative coordinate GPS, mentre nel secondo si esaminerà il testo per cercare concetti rilevanti, Roma in questo scenario. Dalla Figura 4.2 emergono, inoltre, altri aspetti salienti relativi alla natura di un post. Si noti, infatti, come il tag geolocalizzato ad esso collegato presenti, oltre al nome del punto di interesse che si vuole condividere, ulteriori metadati, quali la città dove questi si trova, la categoria di afferenza (luogo storico nel caso del Colosseo), il numero di altri amici che vi hanno effettuato un checkin.

[4]https://developers.facebook.com/docs/graph-api

Figura 4.2: Post di un utente sul suo diario Facebook. Il testo si trova sotto al nome e alla foto del profilo. Nel post "Quanto sei bella Roma!!!" è stato assegnato il tag geolocalizzato "presso Colosseo": esso è relativo al POI Colosseo, che appartiene alla categoria *luogo storico*, si trova a Roma ed è già stato visitato da altri 27 amici dell'utente.

4.4 Tag geolocalizzati

Un aspetto cruciale dei tag geolocalizzati riguarda la totale libertà lasciata da Facebook nella loro scelta all'interno del post di un utente. Infatti, per aggiungere un luogo al proprio messaggio o alla propria foto, un utente può immettere un termine nella form di ricerca; il social network gli restituirà una lista di posti che corrispondono alla chiave digitata. Ciascuno di questi risultati è un *Facebook Place* e vi è associata una *Facebook Page*. Ad esempio, la pagina relativa al Colosseo contiene una foto del monumento, una sua breve descrizione, informazioni geografiche (latitudine e longitudine, indirizzo, città, nazione), categorie che è possibile associare al concetto (in questo caso luogo storico, monumento, attrazione pubblica), link del Colosseo su Wikipedia, persone a cui piace la pagina.

Tali *Facebook Page*, tuttavia, non sono ufficiali. Esse sono inserite manualmente e gestite da un utente della comunità, senza particolari controlli operati dagli amministratori del social network, il che determina problemi non solo di inconsistenza dei dati ma anche di duplicazione di pagine. Una persona che ha

intenzione di convidere sul proprio profilo la posizione attuale, ad esempio lo studentato *Moholt*, nella città di Trondheim, in Norvegia, potrà scegliere tra vari elementi della lista, come "Moholt Studentby, Trondheim", "Moholt stud.by", "Moholt Trondheim!", "Moholt", come si evince dalla Figura 4.3. Si delinea, pertanto, una sfida complessa per un sistema il cui obiettivo è estrarre ed analizzare le attività di un utente. Tutti gli elementi della lista precedente, infatti, sono considerati da Facebook come scorrelati tra loro in quanto sintatticamente non equivalenti; in realtà, esse rappresentano semanticamente lo stesso concetto, ovvero lo studentato Moholt, all'indirizzo Herman Krags Vei, nella città di Trondheim, in Norvegia.

Altro aspetto rilevante da considerare è che la libertà nella scelta dei tag da poter assegnare a un post provoca una proliferazione di tag con chiavi poco significative che causano difficoltà nell'identificazione da parte di un sistema di estrazione. Inoltre, tali annotazioni risultano inefficaci per la modellazione di una comunità di utenti, come in Cicero, a causa della loro ridotta frequenza di riutilizzo. Un esempio incontrato spesso durante la sperimentazione è il tag "A casa mia!": esso è difatti legato a un luogo personale, soggettivo e dotato di coordinate GPS non univoche, variabili da persona a persona.

4.5 Algoritmo di disambiguazione di tag geolocalizzati

A partire dalle considerazioni presentate nel paragrafo precedente, sorge la necessità di dotare un sistema di modellazione utente che sfrutta tag geolocalizzati della funzionalità di disambiguare i concetti. Da un lato, infatti, le annotazioni sintatticamente differenti devono essere unite tra loro se semanticamente rappresentano lo stesso concetto; dall'altro vanno escluse quelle troppo soggettive e di scarso interesse collettivo.

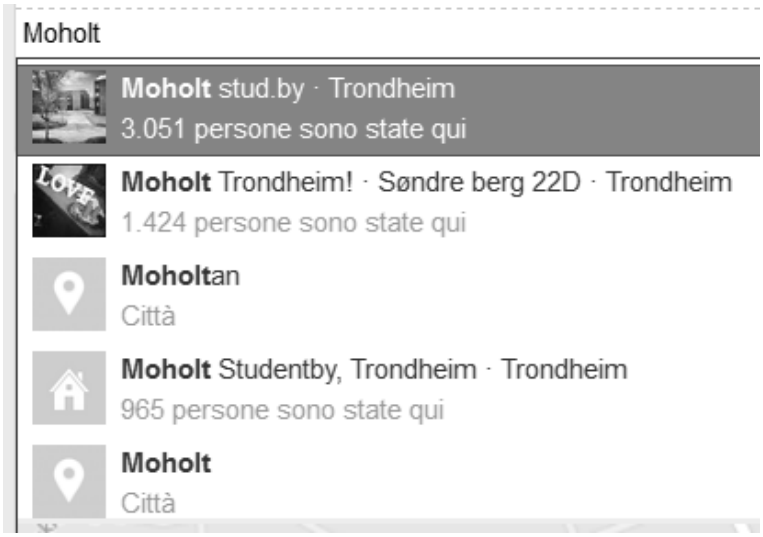

Figura 4.3: Lista di pagine restituite per associare il tag di un luogo a un post su Facebook.

L'algoritmo di disambiguazione proposto dal sistema Cicero sfrutta i Linked Open Data e due servizi di Named Entity Recognition presenti in letteratura, ovvero *GeoNames* e *TagMe*. GeoNames è un database geografico che include tutte le nazioni del pianeta e contiene oltre otto milioni di nomi di luoghi disponibili e scaricabili gratuitamente. Può essere interrogato tramite opportune API[5] con client sviluppati per i principali linguaggi di programmazioneo o sottoponendo query direttamente agli endpoint REST che GeoNames espone. Punto di forza di questo servizio è la possibilità di recuperare luoghi geografici non solo in base al nome, ma anche in base alla vicinanza a determinate coordinate di latitudine e longitudine passate come parametri di input. TagMe, invece, è un potente tool in grado di identificare a runtime delle brevi frasi di testo strutturate e ricche di significato, collegandole a delle pagine di Wikipedia pertinenti ad esse, in modo veloce ed efficace. Essendo un progetto dell'Università di Pisa, TagMe è uno dei pochi strumenti di Named Entity Recognition allo Stato dell'Arte con supporto nativo in lingua italiana, risultando così un validissimo alleato nel processo di

[5]www.geonames.org/export/web-services.html

disambiguazione del sistema oggetto della tesi.

L'input dell'algoritmo di disambiguazione adottato in Cicero è un località estratta da un post o da una foto dell'utente, costituita da un nome (la stringa impiegata nel tag, ad esempio Moholt stud.by) e dalle coordinate GPS del tag, espresse nella forma di latitudine e langitudine. L'output è un'entità identificata univocamente da un URI DBpedia, al quale saranno associate tutte quelle annotazioni che semanticamente rappresentano quel determinato concetto, pur presentando evidenti differenze a livello sintattico. La funzione mapperà così le stringhe "Moholt Studentby, Trondheim", "Moholt stud.by", "Moholt Trondheim!", "Moholt" sull'URI *http://dbpedia.org/resource/Moholt.*

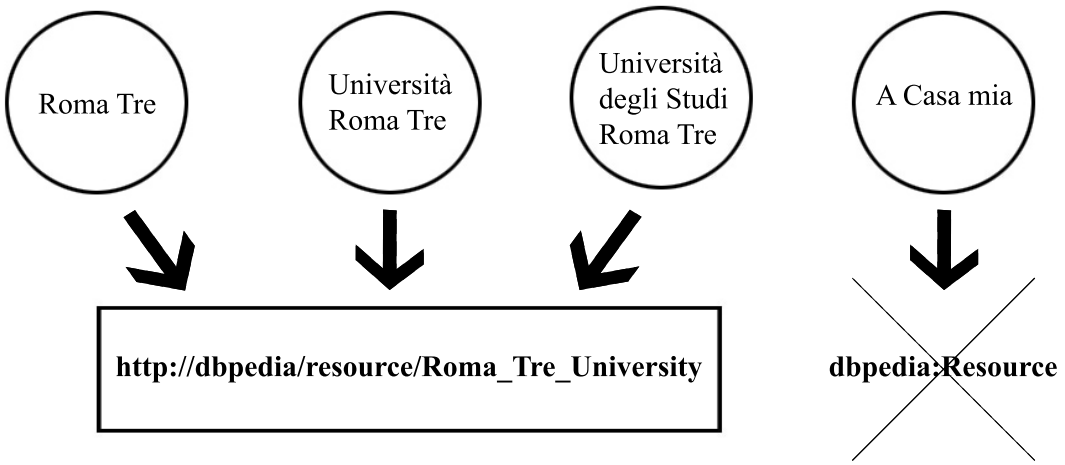

Figura 4.4: Esempio di mapping di risorse effettuato dall'algoritmo di disambiguazione proposto in Cicero. Le stringhe "Roma Tre", "Università Roma Tre" e "Università degli Studi Roma Tre" sono associate all'URI dbpedia:Roma_Tre_University. Il tag "A Casa mia", non avendo corrispondenze rilevanti nell'ontologia dbpedia:Resource di riferimento, non viene disambiguato; pertanto questi non sarà associato ad alcun URI.

In una prima fase il nome del posto da disambiguare viene sottoposto a TagMe, mediante sia l'endpoint italiano che quello inglese. Questo servizio restituirà una lista di elementi che si ritiene possano essere un match valido per il termine di ricerca. Ciascun concetto è identificato dall'URI di una risorsa di DBpedia.

Tuttavia, trattandosi di stringhe molto corte, spesso termini singoli, è spesso difficile per TagMe comprendere appieno il contesto e disambiguare correttamente in presenza di omonimia. Si consideri lo scenario in cui in input ci sia il testo costituito dal solo termine "Nettuno": l'utente stava pensando a Nettuno inteso come città, come dio del mare secondo la religione degli antichi romani o come ristorante Nettuno di una località marittima? TagMe non ha a disposizione abbastanza dati per prendere una decisione. A tal proposito, le informazioni geolocalizzate incluse nel tag sociale dello stream di Facebook si rivelano fondamentali per l'algoritmo qui proposto. A partire dall'URI, viene sottoposta una query SPARQL all'endpoint di DBpedia, per restituire eventualmente latitudine e longitudine della risorsa RDF. Qualora presenti, tali valori vengono confrontati con le coordinate GPS del tag: se la distanza di Manhattan tra i due punti è inferiore ai 500m allora è alta la probabilità che il match sia perfetto. Nel caso vengano riconosciuti più candidati, si considera migliore quello che minimizzi tale distanza. Infatti, minore è la distanza tag-POI, maggiore è la probabilità che tale tag sia effettivamente relativo a quel POI.

Con *distanza di Manhattan* (*Manhattan Distance* o *Taxicab geometry*) si intende la distanza tra due punti calcolata come somma del valore assoluto delle differenze delle loro coordinate. Essa è ricavabile con la Formula 4.1, dove P_1 è un punto di coordinate (x_1, y_1) e P_2 è un punto di coordinate (x_2, y_2). Nel caso dell'algoritmo di disambiguazione, x e y corrispondono, rispettivamente, alla latitudine e alla longitudine del tag.

$$MD(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2| \tag{4.1}$$

La distanza di Manhattan può essere calcolata anche per due generici punti P di più dimensioni, con $P \in R^n$.

A volte, tuttavia, TagMe non è in grado di rilevare potenziali concetti per un dato testo di input, restituendo un output nullo. Al verificarsi di tale situazione, l'algoritmo effettuea un tentativo di disambiguazione tramite il servizio GeoNames. Le sue API permettono varie famiglie di interrogazioni. In particolare, in

Cicero si sfruttano le query testuali di GeoNames, similmente a quanto avviene con TagMe, e le query geografiche: vengono cioè restituiti posti dal database di GeoNames le cui coordinate siano vicine a quelle fornite in input. Anche i risultati delle ricerche testuali di GeoNames vengono filtrati in base alla distanza di Manhattan dal punto definito da latitudine e longitudine in cui è stato assegnato il tag su Facebook.

Nel caso in cui anche la lista dei risultati di GeoNames sia vuota, allora il posto rappresentato dal tag viene escluso. Come si discuterà nei prossimi paragrafi, in realtà non è completamente ignorato: vengono comunque memorizzate le categorie della pagina Facebook di cui il tag è parte. Tali attributi, difatti, consentono al sistema di comprendere in modo più approfondito le preferenze dell'utente.

Caratteristica importante da notare è che sia GeoNames che TagMe identificano le entità in base all'URI della risorsa RDF del dataset semantico di DBpedia.

Concetti di scarso interesse collettivo, come "A casa mia!", a cui non è associata alcuna risorsa RDF in tale dataset, sono da essi ignorati automaticamente. Il framework può quindi rappresentare l'attività utente attraverso i metadati e le risorse RDF ad essa correlate.

Oltre ai tag esplicitamente aggiunti da un account di Facebook, nella prima fase Cicero analizza anche i titoli testuali dei post e delle foto estratti, per estrarre le annotazioni implicite. Analogamente ai tag espliciti, il framework si avvale del servizio di Named Entity Recognition TagMe. Tuttavia, non essendoci dati geolocalizzati annessi al messaggio, la lista dei risultati non viene filtrata per distanza di Manhattan ma si sceglie, in questo caso, il concetto che, secondo TagMe, è il più inerente al contesto.

Tabella 4.1: Tabella che elenca alcuni risultati ottenuti dall'algoritmo di disambiguazione proposto in Cicero. La prima colonna contiene la stringa di testo relativa al nome del luogo così come è presente all'interno di un tag Facebook. La seconda e la terza sono costituite dall'output dell'operazione di disambiguazione, rispettivamente dal nome e dall'URI DBpedia della risorsa su cui l'algoritmo ha mappato ciascun tag fornito in input.

Nome del luogo	**Nome dopo disambiguazione**	**URI DBpedia risorsa**
Università Roma Tre	Roma Tre University	dbpedia:Roma_Tre_University
Montesilvano	Montesilvano	dbpedia:Montesilvano
Ariccia	Ariccia	dbpedia:Ariccia
Rome Italy	Rome	dbpedia:Rome
Oxford Ohio	Oxford Ohio	dbpedia:Oxford,_Ohio
South Africa	South Africa	dbpedia:South_Africa
Moholt stud.by	Moholt	dbpedia:Moholt
Piazza Navona	Piazza Navona	dbpedia:Piazza_Navona
Trondheim Norway	Trondheim	dbpedia:Trondheim
Biberach Baden-Wurttemberg	Biberach (district)	dbpedia:Biberach_(district)
Morelia Mexico	Morelia	dbpedia:Morelia
Gernika	Guernica	dbpedia:Guernica
Museo Guggenheim Bilbao	Guggenheim Museum Bilbao	dbpedia:Guggenheim_Museum_Bilbao
Vatican Museums - Musei Vaticani	Vatican Museums	dbpedia:Vatican_Museums
Playa Maspalomas	Maspalomas	dbpedia:Maspalomas

4.6 Grafo sociale e Neo4j

Una volta che i dati utente sono stati estratti da Facebook diventa importante renderli persistenti in modo che possano essere interrogati facilmente e con alte prestazioni per le successive operazioni di modellazione e raccomandazione. La natura dei social network ci suggerisce metodi alternativi al salvataggio dei dati nel tradizionale modello relazionale. Dove un RDBMS è ottimizzato per dati aggregati, *Neo4j* lo è per risorse altamente connesse come quelle di un grafo sociale. La struttura tabellare sarebbe infatti inefficiente in questo contesto: richiederebbe un eccessivo uso di *join* per la navigazione dei nodi della rete. Al contrario un *graph database* permette di salvare i concetti di vertici e archi in modo assolutamente naturale ed elegante.

La Figura 4.5 mostra come sia possibile modellare un dominio in modo

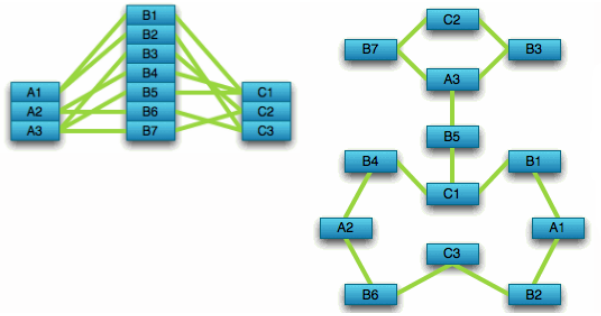

Figura 4.5: Conversione di uno schema relazionale in un corrispondente modello a grafo, proprio di un graph db.

equivalente sia nella struttura relazionale, che in quella a grafo.

4.6.1 Il grafo sociale in Cicero

Il grafo sociale proposto in Cicero è orientato, etichettato ed eterogeneo, nel senso che ciascun nodo è caratterizzato da una propria semantica. Esso è composto da quattro diverse classi di nodi (*Person*, *Place*, *Location*, *Category*) e quattro possibili label per gli archi (*KNOWS*, *VISITED*, *LOCATED_IN*, *HAS_CATEGORY*).

Person: una persona con un account attivo sul social network. Vi appartengono l'utente che si vuole modellare e i suoi amici;

Place: indica un qualsiasi posto fisico visitato. Viene identificato tramite la disambiguazione del tag geolocalizzato, oppure è interpretato da un messaggio di testo tramite operazioni di Named Entity Recognition;

Location: è un nodo che contiene le informazioni geografiche di un luogo. E' disaccoppiato dal concetto di Place perchè alcuni POI, come musei o attività commerciali, non sono sempre legati a un punto fisso ma potrebbero variare indirizzo nel tempo. La Figura 4.6 riporta un esempio di nodo Location;

Category: rappresenta la categoria del posto, derivata da quella della Facebook Page corrispondente. Al momento della stesura della tesi, il social network prevede un totale di 238 categorie da cui scegliere;

KNOWS: relazione tra due nodi Person che sono amici tra loro su Facebook;

VISITED: relazione tra Person e Place: l'utente ha visitato un luogo e lo ha condiviso su Facebook tramite un tag geolocalizzato;

LOCATED_IN: arco che unisce un posto con la sua locazione al momento del tag;

HAS_CATEGORY: arco diretto da un nodo Place a un nodo Category. La label è esplicativa: rappresenta l'appartenenza di un POI a una data categoria.

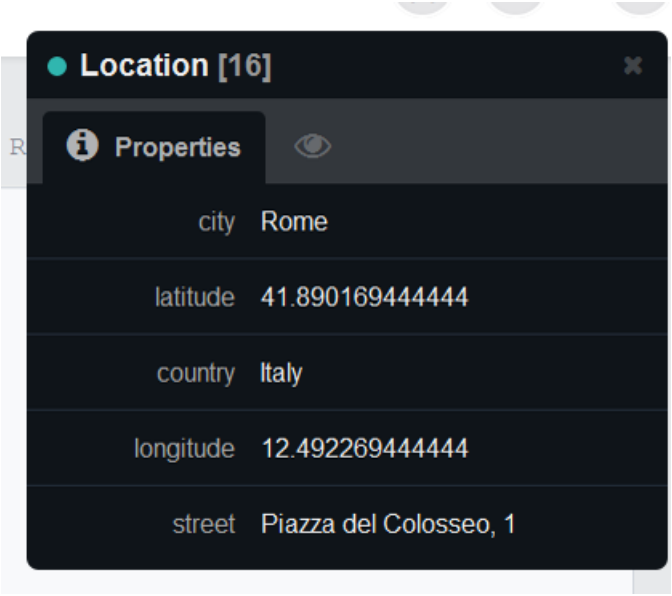

Figura 4.6: Proprietà di un nodo Location appartenente al grafo sociale di un utente di Cicero. Tra di esse si annoverano la città, la latidudine, la longitudine, l'indirizzo e la nazione.

A partire dall'intera rete sociale di Facebook, il grafo relativo a ciascun utente considerato all'interno di Cicero è un *cluster* di essa. Il nodo centrale, la sorgente, è costituita dal nodo Person che identifica l'utente del sistema. Ciascun cluster è distinto e rappresenta la communità composta dall'utente stesso e da tutti i suoi amici di Facebook. In Cicero è perciò presente un grafo (e quindi un corrispondente database Neo4j) diverso per ogni utilizzatore del sistema. La Figura 4.7 mostra un esempio di grafo sociale di un utente di Cicero reso persistente su Neo4j. Come si evince dalla legenda annessa, ogni tipologia di nodo è colorata diversamente.

4.7 Raccomandatori proposti in Cicero

Vari raccomandatori vengono proposte nel sistema Cicero e in questo paragrafo ne sono delineati i tratti salienti. Possono essere partizionati in due gruppi principali: *raccomandatori sociali* e *raccomandatori semantici*. Caratteristica importante del framework è la possibilità di unire in sequenza più recommender per la formazione di strategie combinate, al fine di sfruttare i loro vantaggi in modo più accurato.

4.7.1 Raccomandatori sociali

L'obiettivo dei raccomandatori sociali è analizzare i dati provenienti dalle attività dell'utente del sistema su Facebook, precedentemente estratti e resi persistenti nel graph database. Tali algoritmi modellano il profilo utente, cercando di cogliere le sue preferenze per consigliargli una lista di oggetti Place. I raccomandatori sociali di Cicero adottano un filtraggio *Community-Based*, discusso nel precedente capitolo, considerando la comunità costituita dall'insieme di tutti gli amici Facebook dell'utente target. Nel seguito, sono presentati il *Random Social Recommender*, *Pure Community-Based Social Recommender* e il *Collaborative Filtering Social Recommender*.

Figura 4.7: Esempio di grafo sociale di un utente di Cicero reso persistente su Neo4j.

4.7.1.1 Random Social Recommender

Il Random Social Recommender si profila come un raccomandatore naïve. Dei posti visitati dalla comunità, ne vengono restituiti alcuni in modo casuale. Il fondamento su cui si basa il suo approccio di filtraggio nello spazio degli item è che, se almeno un proprio amico ha visitato un luogo, allora potrebbe risultare interessante all'utente stesso. Esso non è particolarmente elaborato: è presente all'interno del sistema per consentire confronti con approcci più sofisticati e allo Stato dell'Arte, valutando se tali tecniche di filtraggio risultino effettivamente più

efficaci quando la struttura del dataset è a grafo.

4.7.1.2 Inferire le categorie preferite dell'utente

Il *Pure Community-Based Social Recommender* e il *Collaborative Filtering Social Recommender*, come primo passo, profilano le preferenze dell'utente in termini di categorie di posti interessanti. Di tutti i node Place che presentano un link VISITED con il nodo Person che caratterizza l'utente, vengono restituiti i vertici di tipo Category ad essi connessi, ordinati in ordine decrescente in base allo score. Questo punteggio corrisponde al numero di link HAS_CATEGORY entranti in tali vertici. L'output viene successivamente filtrato in base alla seguente strategia. Delle 238 categorie totali di Facebook ne sono state selezionate 37 inerenti al dominio dei beni culturali, come *City*, *Monument*, *Museum*, *Historical Place*, *Touristic Attraction* e *Church*. Qualora la categoria estratta dal graph database dell'utente appartenga a tale insieme, il punteggio iniziale relativo riceve un *boost*. Sia x lo score della categoria, c la categoria, α un parametro tale che $0.5 < \alpha < 1$. Allora x può essere calcolata con la Formula 4.2.

$$x = \begin{cases} x * \alpha & \text{if c} \in \text{CulturalHeritageCategories} \\ -x * (1 - \alpha) & \text{if c} \notin \text{CulturalHeritageCategories} \end{cases} \tag{4.2}$$

Risultati empirici hanno mostrato che 0.85 sia il valore migliore da attribuire ad α. La scelta $\alpha = 1$ eliminerebbe totalmente il contributo delle categorie meno afferenti al dominio dei beni culturali, con la conseguenza di perdere caratteristiche rilevanti dei gusti dell'utente. La scelta $\alpha = 0.5$ attribuirebbe invece uno stesso peso a tutte le categorie, senza distinguere se siano relative o meno a un bene culturale. Lo score viene successivamente normalizzato a 1, dividendo ognuno di essi per il massimo della lista e, se inferiori alla media, tali punteggi vengono filtrati. Di tutte le categorie rimaste si scelgono infine le top five.

Da notare che, all'interno dell'algoritmo, vengono analizzati anche i link tra le categorie ed i POI che la procedura di disambiguazione non è riuscita a riconoscenere. Tali posti, infatti, pur non compresi nelle liste di item raccomandati dal sistema, forniscono informazioni importanti circa gli interessi dell'utente, soprattutto in termini di preferenze sul tipo di luoghi che generalmente visita.

4.7.1.3 Pure Community-Based Social Recommender

Come si evince dal nome, tale recommender presenta un approccio totalmente community-based. Il presupposto principale è che un amico sia una persona di cui si possa fidare e che, quindi, faccia parte della propria *Trust Network*. E' altamente probabile che i posti più popolari tra gli amici risultino di interesse anche per l'utente. L'algoritmo prevede l'interrogazione del grafo sociale dell'utente per recuperare tutti i nodi di tipo Place che appartengono alle categorie precedentemente calcolate, ordinati in ordine descrescente in base al numero di link "VISITED" entranti, ovvero le persone che li hanno visitati. Lo score di preferenza per un POI è quindi assegnato tramite un peso basato sulle occorrenze. Infine la quantità di item restituiti viene limitata ai migliori dieci.

```
MATCH (friend:Person)-[visited:VISITED]->(place),
       (place)-[:HAS_CATEGORY]->(favouriteUserCategory:Category)
WHERE favouriteUserCategory.name = {categoryName}
RETURN distinct place, count(visited) as visitors
ORDER BY visitors DESC;
```

Figura 4.8: Query espressa nel linguaggio Cypher per filtrare, dal grafo di Neo4j, i posti di una data categoria visitati dall'utente di Facebook e dalla sua comunità di amici tramite l'approccio comunitario introdotto nel Pure Community Based Social Recommender.

4.7.1.4 Collaborative Filtering Social Recommender

All'interno della comunità degli amici non è sempre detto che gli interessi di tutti gli amici corrispondano ai propri. Il *Collaborative Filtering Social Recommender* prevede tecniche di filtraggio proprie dell'approccio collaborativo. Il fine è selezionare le persone che abbiano i gusti più simili a quelli dell'utente, restituendo, all'interno del grafo sociale, solo gli item ad esse connessi. L'output dell'algoritmo è quindi costituito da tutti quei nodi Place, appartenenti alle categorie preferite, dove si sono recati quegli amici che sono stati taggati in posti visitati dall'utente stesso e presentano interessi affini. La lista risultante è ordinata in base al numero di archi entranti ai Place con label "VISITED".

```
MATCH (user:Person)-[:VISITED]->()
       <-[:VISITED]-(friend:Person)-[:VISITED]->(place),
       (place)-[:HAS_CATEGORY]->(category:Category)
WHERE user.id = {userId} AND favouriteUserCategory.name = {categoryName}
RETURN distinct place, count(*) as visitors
ORDER BY visitors DESC;
```

Figura 4.9: Query espressa nel linguaggio Cypher per filtrare, dal grafo di Neo4j, i posti di una data categoria visitati dall'utente di Facebook e dalla sua comunità di amici tramite l'approccio collaborativo introdotto nel Collaborative Filtering Social Recommender.

4.7.1.5 Cold-start e sparsità degli item

Quello del cold-start è un problema di cui soffrono frequentemente i sistemi di raccomandazione. Esso consiste nel non avere a disposizione uno storico sufficiente di dati con cui produrre suggerimenti per un utente, evento che occorre in particolare quando questi si è da poco registrato al servizio. Cicero tenta di contrastare il cold-start adottando un approccio community-based. Anche se l'utente ha recentemente aperto il proprio account sul social network o non ha ancora assegnato un tag geolocalizzato ad alcun post, il framework è comunque

in grado di avere dei concetti iniziali da cui partire, esplorando gli interessi della comunità di amici. Il problema sussiste, tuttavia, qualora l'utente abbia stretto un numero esiguo di amicizie o se anche i suoi amici non sono particolarmente attivi sulla piattaforma, in termini di risorse condivise. E' stato scelto Facebook come sorgente proprio per il fatto di essere il social network online correntemente più diffuso ed usato globalmente. Ciò aiuta a ridurre l'occorrenza del cold start durante il normale utilizzo di Cicero.

Un discorso simile può essere esteso anche all'algoritmo di inferenza delle categorie di punti di interesse preferiti dall'utente discusso precedentemente. Esso, infatti, non sempre possiede sufficienti informazioni. All'occorrenza di tale situazione vengono considerate come di rilievo le categorie più visitate dal proprio cluster di amici, adottando nuovamente un approccio community-based.

Analogamente al cold-start, la sparsità degli item è una questione da non sottovalutare nella progettazione di un Recommendation System. In Cicero la fase di disambiguazione dei tag attenua fortemente questo fenomeno. Si consideri nuovamente lo scenario relativo allo studentato Moholt e che quattro amici vi abbiano condiviso la posizione adoperando ciascuno un tag differente: "Moholt Studentby, Trondheim", "Moholt stud.by", "Moholt Trondheim!", "Moholt". Senza un'operazione di disambiguazione, nel graph database si salverebbero le seguenti triple secondo la notazione di Neo4j, ciascuna di peso 1 (un solo arco VISITED entrante):

```
(Friend1)-[VISITED]->(Moholt Studentby, Trondheim),
(Friend2)-[VISITED]->(Moholt stud.by),
(Friend3)-[VISITED]->(Moholt Trondheim!),
(Friend4)-[VISITED]->(Moholt).
```

Riconoscendo tali tag come afferenti alla stessa entità, *dbpedia:Moholt*[6], si avrà

[6]Il prefisso *dbpedia:* è l'abbreviazione usata nel mondo LOD per indicare tutte le risorse afferenti al namespace http://dbpedia.org/resource/. L'URI dell'entità Moholt, per esteso, è pertanto http://dbpedia.org/resource/Moholt. Ciascun namespace possiede un proprio prefisso identificativo, univoco all'interno del Semantic Web.

un solo nodo Place *dbpedia:Moholt* di peso 4 (quattro archi VISITED entranti), con un conseguente abbattimento della sparsità. Tale tecnica, unita al filtraggio degli item di scarso interesse collettivo (tag "A casa mia!"), riduce drasticamente il numero di nodi che compongono il grafo sociale su Neo4j.

4.7.2 Raccomandatori semantici

I raccomandatori semantici arricchiscono la modellazione utente con informazioni di background provenienti dalle sorgenti dei Linked Open Data, in particolare dai progetti DBpedia ed Europeana. In Cicero sono proposte varie tecniche: *DBpedia Closer Places Recommender* e *Europeana Recommender*.

4.7.2.1 DBpedia Closer Places Recommender

L'idea sui cui si fonda *DBpedia Closer Places Recommender* è che, se un utente ha apprezzato un posto talmente da volerlo condividere su un social network, è altamente probabile che sia interessato anche ad altri punti di interesse che si trovino nelle vicinanze del luogo in cui è stato effettuato il tag geolocalizzato e che siano simili ad esso da un punto di vista semantico.

Questo raccomandatore sfrutta la base di conoscenza del LOD di DBpedia. A partire da una lista di URI di risorse DBpedia, DBpedia Closer Places Recommender consiglia all'utente concetti che condividano categorie semantiche (sotto forma della proprietà *dcterms:subject* definita dal namespace *http://purl.org/dc/terms/*) con le risorse di input e che non siano particolarmente distanti da essi. Si tende così a privilegiare quei beni artistici più facilmente raggiungibili dall'utente. Le interrogazioni di cui si avvale DBpedia Closer Places Recommender sono formulate in SPARQL, come mostrato dal codice presente nella Figura 4.10, in cui il concetto di input è rappresentato dall'URI *dbpedia:Colosseum*. Per migliorare la diversità degli item raccomandati vengono inseriti nella lista di output al massimo quattro item semanticamente simili per ogni URI di input.

Per migliorare l'esperienza utente, il sistema è in grado di fornire non solo il titolo dell'opera ma anche un'immagine rappresentativa e un link esterno che reindirizza verso la corrispondente pagina Wikipedia. Tali metadati sono infatti facilmente estraibili dal dataset RDF.

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT distinct ?concept
WHERE {
    dbpedia:Colosseum  geo:lat ?uriLat.
    dbpedia:Colosseum  geo:long ?uriLong.
    dbpedia:Colosseum dcterms:subject ?sub.
    ?concept geo:lat ?lat.
    ?concept geo:long ?long.
    ?concept dcterms:subject ?sub.
    FILTER(?lat - ?uriLat <= 0.05 && ?uriLat - ?lat <= 0.05 &&
       ?long - ?uriLong <= 0.05 && ?uriLong - ?long <= 0.05 &&
       ?concept!=dbpedia:Colosseum).
}
LIMIT 10;
```

Figura 4.10: Esempio di query SPARQL formulata da DBpedia Closer Places Recommender: interroga il dataset di DBpedia per cercare quei concetti che condividono categorie (*dcterms:subject*) con la risorsa *dbpedia:Colosseum* e che non distano più di 5km da essa.

4.7.2.2 Europeana Recommender

Europeana Recommender esplora i grafi RDF forniti dalla collezione di Europeana per fornire suggerimenti di risorse (quali libri, film, dipinti, giornali, video, audio, mappe, manoscritti, ...) che abbiano attinenza con i gusti dell'utente. L'input è una lista di posti di cui è fornito il nome, non necessariamente dotati di un URI semantico annesso. Per ogni elemento vengono recuperati dalla colle-

zione e raccomandate all'utente quelle opere il cui titolo o il cui soggetto da esse rappresentato coincide, per tutta la lunghezza della stringa o solo parzialmente, con il suo nome. Altri metadati sono estratti oltre al titolo e al soggetto: il creatore dell'opera, l'URL del media della risorsa sotto forma di foto o video o di *thumbnail* (in caso di documenti, libri o manoscritti) audio e un link esterno che rimanda alla rispettiva pagina del sito web di Europeana, per permettere una contestualizzazione più approfondita dell'opera.

Di conseguenza, la grana degli item estratti è di gran lunga più fine rispetto a quelli che potrebbero essere recuperati da un dataset DBpedia. Si consideri a titolo esemplificativo il *Colosseo* e una relativa interrogazione sulle due sorgenti LOD. Mentre DBpedia restituisce la risorsa dbpedia:Colosseo che descrive il momumento romano in quanto tale, Europeana fornisce singole rappresentazioni di esso, come foto, dipinti, video, contenuti all'interno di musei, mostre o collezioni europeee.

4.7.2.3 SemanticSimilarityDBpediaRecommender

Un algoritmo sviluppato in Cicero che, tuttavia, non ha potuto trovare riscontro nella fase di sperimentazione del sistema, per via della sua complessità di elaborazione e dei lunghi tempi di attesa per l'utente, è il *SemanticSimilarityDBpediaRecommender*. Per completezza di trattazione si introduce ugualmente la teoria su cui questo recommender si basa. A partire da un URI di una risorsa DBpedia, corrispondente a un luogo visitato dall'utente, l'algoritmo naviga il dataset di DBpedia per restituire quei concetti che abbiano il maggiore livello di *similarità semantica* con quello fornito in input. La similarità semantica è qui definita come il numero totale di occorrenze di un insieme di graph pattern. Sia c_i il concetto (identificato da un URI) in input, c_o il concetto di cui si vuole computare la similarità semantica con c_i, c_x un nodo qualsiasi del grafo RDF, connesso a c_i, a c_o o ad entrambi, p, p_1, p_2, p_3 sono le possibili proprietà che possono sussistere tra due risorse. I graph pattern, espressi nella forma di tri-

```
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT distinct ?proxy ?creator ?mediaURL ?provider ?title ?source ?cho
WHERE {
      ?s dc:creator ?creator;
         ore:proxyIn ?proxy;
         dc:subject ?subject;
         dc:title ?title;
         dc:title ?source;
         dc:type ?type.
      ?proxy edm:isShownBy ?mediaURL.
      ?proxy edm:dataProvider ?provider.
      ?proxy edm:aggregatedCHO ?cho.
      {    ?s dc:title "Trondheim".}
       UNION
      {     ?s dc:subject "Trondheim" .}
}
LIMIT 100;
```

Figura 4.11: Esempio di query SPARQL formulata da Europeana Recommender: interroga il database di Europeana per estrarre tutte quelle opere in esso presenti che abbiano la stringa "Trondheim" come titolo o soggetto. L'output è costituito dalla risorsa e i suoi metadati (e.g., titolo, soggetto, media, creatore, tipo, fornitore).

ple RDF *soggetto-predicato-oggetto*, che possono dunque occorrere nel dataset e contribuire al grado di similarità, sono i seguenti:

- $c_i\ p\ c_o$.
- $c_o\ p\ c_i$.
- $c_o\ p_1\ c_x.\ c_i\ p_2\ c_x$.
- $c_o\ p_1\ c_x.\ c_x\ p_2\ c_i$.

- $c_i\ p_1\ c_x.\ c_x\ p_2\ c_o.$
- $c_x\ p_1\ c_x.\ c_x\ p_2\ c_i.$
- $c_o\ p_1\ c_o.\ c_o\ p_2\ c_x.\ c_x\ p_3\ c_i.$
- $c_i\ p_1\ c_o.\ c_o\ p_2\ c_x.\ c_x\ p_3\ c_o.$

I primi due sono pattern diretti, in cui cioè c_i e c_o appartengono allo stesso statement RDF. I restanti sono invece indiretti: è possibile raggiungere c_o da c_i attraverso una navigazione del grafo ad un livello di profondità maggiore. L'output dell'algoritmo *SemanticSimilarityDBpediaRecommender* è una lista di coppie $< c_i,\ c_o >$ ordinate in modo decrescente in base al loro valore di similarità semantica.

Non sempre c_o risulta incognito a priori. Spesso è un elemento della lista di item raccomandati da un precedente algoritmo, soprattutto nelle situazioni in cui più raccomandatori vengano concatenati in sequenza per generare i suggerimenti. In questo scenario si vuole calcolare il grado di similarità semantica che sussiste tra due item del grafo sociale, inizialmente correlati solo dal fatto di essere entrambi luoghi visitati dalla comunità degli amici dell'utente, per permettere al sistema di raccomandare quelle risorse di DBpedia che siano maggiormente vicine, da un punto di vista semantico, a un luogo di interesse per l'utente. Anche in questo caso l'output di *SemanticSimilarityDBpediaRecommender* è una lista di coppie $< c_i,\ c_o >$ ordinate per similarità semantica.

4.7.3 Pipeline di raccomandatori

Un punto di forza degli algoritmi di raccomandazione proposti in Cicero è la possibilità fornita all'utente di combinarli tra loro in pipeline, al fine di avere risultati più accurati e variegati, sfruttando appieno i singoli vantaggi di ognuno di essi. Si possono adoperare raccomandatori sociali da soli o combinati con altri semantici. All'interno della pipeline l'output di un raccomandatore si trasforma

CICERO THINKS YOU MAY LIKE Lade, Trondheim SO HE RECOMMENDS YOU

TITLE	Ladestien
PROVIDER	DBPEDIA
MEDIA	
LINK	http://en.wikipedia.org/wiki/Ladestien

This item matches my interests

- Strongly Agree
- Agree
- Neither Agree Nor Disagree
- Disagree
- Strongly Disagree

CICERO THINKS YOU MAY LIKE Trondheim SO HE RECOMMENDS YOU

TITLE	Sverresborg
PROVIDER	DBPEDIA
MEDIA	
LINK	http://en.wikipedia.org/wiki/Sverresborg

This item matches my interests

- Strongly Agree
- Agree
- Neither Agree Nor Disagree
- Disagree
- Strongly Disagree

Figura 4.12: Esempio di risultato generato dalla pipeline *Collaborative Filtering + DBpedia Closer Places*.

Figura 4.13: Esempio di item raccomandato dalla pipeline *Community Based + Europeana*.

nell'input del successivo. Alcune configurazioni possibili, ad esempio, sono *Social Recommender + DBpedia Closer Places Recommender, Social Recommender + Europeana Recommender oppure Social Recommender, Social Recommender + DBpedia Closer Places + Europeana Recommender*, dove con *Social Recommender* si intende uno qualsiasi dei raccomandatori sociali illustrati.

Le Figure 4.12 e 4.13 mostrano due esempi di liste di item raccomandati, generate da due pipeline differenti. Una è prodotta da *Collaborative Filtering + DBpedia Closer Places*, l'altra da *Community Based + Europeana*. Si noti come il sistema notifichi all'utente il motivo per cui è stata suggerita quella determinata risorsa: ad esempio "Cicero thinks you may like *Trondheim*, so he recommends you *Sverresborg*.". All'interno della pipeline il contributo fornito dal *Collaborative Filtering Recommender* è quello di filtrare, nell'intero spazio degli item dell'utente, i posti visitati dalla comunità degli amici che rispecchino maggior-

mente i suoi interessi. A partire dalla risorsa così consigliata, ovvero la città di Trondheim, tramite un'arricchimento semantico effettuato attraverso l'algoritmo *DBpedia Closer Places*, il sistema raccomanderà all'utente il punto di interesse di Sverresborg. Sverresborg (*dbpedia:Sverresborg*), infatti, si trova ai confini della città di Trondheim (*dbpedia:Trondheim*) ed è un museo folkloristico all'aria aperta che mostra, ai suoi visitatori, oltre 80 ricostruzioni storiche delle case di legno dei principali villaggi ottocenteschi della Norvegia.

Accanto a ciascun item restituito, inoltre, è presente anche un'affermazione "*The item matches my interests*", con cinque possibili risposte che consentono all'utente di esprimere un rating esplicito relativamente all'oggetto raccomandato, sulla base di una scala di Likert a 5 valori. Per maggiori dettagli si rimanda al Capitolo 6. I voti totali collezionati vengono successivamente esaminati con opportune metriche statistiche per valutare la bontà del sistema di raccomandazione.

Capitolo 5

Caratteristiche architetturali del sistema Cicero

In letteratura sono state proposte molteplici definizioni di cosa si intende per architettura di un sistema software. In [BCK13] viene presentata nel seguente modo:

> *"The software architecture of a system is the set of structures needed to reason about the system, which comprises software elements, relations among them and properties of both."*

Questo capitolo approfondisce la trattazione del sistema Cicero analizzandone, appunto, l'architettura software: sono esaminati i pattern e le tattiche architetturali adottate, nonchè i motivi per cui sono stati preferiti ad altre alternative; si illustrano i vari componenti e le relazioni che sussistono tra di essi. La modellazione è focalizzata sul punto di vista della "Logic View" del framework, basandosi sulla notazione del *modello 4+1* proposto da Philippe Kruchten in [Kru95]. Essa illustra i moduli da implementare, seguendo un approccio orientato agli oggetti, focalizzandosi in particolar modo sulle funzionalità che il sistema deve fornire all'utente finale.

5.1 Driver architetturali significativi

In questo paragrafo sono illustrati i driver architetturali significativi che caratterizzano Cicero. Essi sono requisiti con un profondo impatto sull'architettura del sistema. La trattazione divide i driver funzionali da quelli non funzionali. I primi stabiliscono cosa il sistema deve fare, quali particolari funzionalità deve possedere, come deve comportarsi o reagire a fronte di determinati stimoli. I secondi concernono non tanto le funzionalità, quanto il soddisfacimento di certi attributi di qualità, quali la modificabilità o la sicurezza, ad esempio.

5.1.1 Driver funzionali

Raccomandazione di beni artistici e culturali: La principale funzionalità di Cicero è la raccomandazione di beni artistici e culturali appartenenti al patrimonio mondiale;

Estrazione informazioni aggiuntive dai Linked Open Data: Il sistema deve essere in grado di estrarre le informazioni semantiche dal Web Of Data, in particolare dalle sorgenti LOD di DBpedia ed Europeana, arricchendo così la modellazione utente con informazioni aggiuntive;

Recupero interessi utente dalle sue attività sui social network: E' necessario dotare il framework di funzionalità che gli consentano di interrogare i social network, Facebook nello specifico. L'obiettivo è analizzare le attività dell'utente e dei suoi amici, in termini di tag geolocalizzati su post e foto da questi condivisi, per inferire i posti visitati e le categorie preferiti;

Memorizzazione del profilo sociale dell'utente: I dati sociali recuperati devono essere resi persistenti in modo che possano essere effettuate successive operazioni di profilazione e raccomandazione;

Fornire giudizi sulla qualità dei raccomandatori: Un utente deve poter formulare un proprio giudizio sulla qualità degli oggetti a lui raccomandati.

5.1.2 Driver non funzionali

Il focus del lavoro qui proposto è sugli attributi di qualità, quali la modificabilità, l'usabilità, le prestazioni e la sicurezza. Questi driver non funzionali influenzano comunque le scelte architetturali.

Modificabilità di raccomandatori e disambiguatori: Il settore di ricerca dei Recommender System è dinamico e in continuo mutamento. Nuove tecniche sono costantemente proposte in letteratura. Cicero deve poter abbracciare il cambiamento e presentare un'architettura che consenta di introdurre con facilità nuovi raccomandatori e disambiguatori.

Modificabilità delle sorgenti dati: Per la profilazione sociale dell'utente si ricorre all'analisi dello stream delle sue attività su Facebook. Il sistema deve essere in grado di supportare l'inserimento di nuovi social network come sorgenti di dati sociali (e.g, *Twitter*, *Google Plus*[1], *Flickr* o *Foursquare*). Discorso analogo è applicabile alle sorgenti LOD da cui si vuole attingere per l'espansione semantica. Anche esse, infatti, possono cambiare nel tempo.

Modificabilità dello strato di persistenza: Per le sue proprietà conformi con i requisiti attuali di Cicero, è stato scelto come strato di persistenza un graph database come Neo4j. Tuttavia il mondo NoSQL è variegato. Potrebbe essere necessario adottare in futuro una diversa categoria di database. Il sistema deve garantire la sostituzione dello strato di persistenza senza dover apportare drastiche modifiche alla sua architettura.

Usabilità (trasparenza per l'utente): Numerosi studi sui RS mostrano come spesso l'esperienza utente sia percepita in modo positivo qualora la fase di training dei raccomandatori avvenga in modo trasparente all'utente. Chi

[1]https://plus.google.com/

ricorre a un RS non vorrebbe difatti dedicare più tempo alla sua configurazione di quanto ne avrebbe speso documentandosi personalmente sugli item del dominio. Nel caso di Cicero l'addestramento avviene con l'estrazione implicita dei dati sociali dell'utente e dei suoi amici per la creazione di un profilo utente iniziale, senza quindi coinvolgere chi usa il sistema. Anche l'operazione di inferenza delle categorie preferite deve risultare trasparente all'utente.

Prestazioni elevate durante la modellazione utente e la raccomandazione: Le chiamate alla GraphAPI di Facebook per l'estrazione dei dati sociali richiedono molto tempo. Inoltre il social network, per garantire una continua disponibilità del servizio a una vasta comunità di utenti che lo possono interrogare da ogni parte del globo, limita la frequenza di chiamate REST (sia singole, sia combinate in batch) per applicazione. E' quindi importante per Cicero ridurre tali richieste per migliorare le proprie prestazioni: l'idea è effettuare le call alle GraphAPI solo una volta per utente, al momento del suo primo login sul sistema, e rendere persistenti i dati estratti. Le successive operazioni di modellazione utente e raccomandazione saranno pertanto più performanti non dovendo interrogare nuovamente gli endpoint di Facebook.

Sicurezza (protezione della privacy): Siccome il framework analizza dati strettamente personali degli utenti e delle loro comunità di amici, è importante salvaguarne la privacy. In conformità con le linee guida delle applicazioni che usufruiscono dei servizi di Facebook, all'utente di Cicero è fornita a priori la possibilità di scegliere le informazioni personali a cui il sistema può accedere. Inoltre deve essere fornita all'utente la possibilità di rimuovere definitivamente i propri dati sociali memorizzati sul graph db di Cicero. Ovviamente, anche il login sull'account di Facebook deve avvenire secondo modalità sicure.

5.2 Tattiche architetturali

Le tattiche architetturali sono scelte di progettazione che influenzano il raggiungimento della risposta di un singolo attributo di qualità quando il sistema è stimolato da un certo evento. Esse differiscono dai pattern nel senso che possono essere viste come un "package" di pattern. In [BCK13] le tattiche architetturali sono trattate approfonditamente per ogni attributo di qualità.

5.2.1 Tattiche per la modificabilità

La *modificabilità* è un attributo fondamentale di un sistema software. Essa concerne il costo del cambiamento e la facilità con cui il sistema può garantire il cambiamento [BBN07]. I cambiamenti avvengono spesso nel corso del tempo e su più fronti, coinvolgendo dunque diversi componenti. In Cicero la modificabilità viene supportata dai pattern architetturali e di design adottati. Il pattern architetturale *multi-tier*, infatti, prevede la separazione delle responsabilità assegnando ciascun compito a un determinato modulo logico facilmente sostituibile. I design pattern adottati, quali *Template Method*, *Facade*, *Repository*, *State*, favoriscono un basso accoppiamento e alta coesione tra le classi, migliorando la modificabilità. Una loro trattazione più dettagliata è presentata nelle prossime sezioni. Altre tattiche impiegate sono la modularità del codice, l'incapsulamento e l'uso di classi intermedie.

5.2.2 Tattiche per le prestazioni

L'attributo di qualità *prestazioni* riguarda il tempo e la capacità del sistema software di soddisfare le tempistiche e i requisiti ad esse annesse. L'obiettivo delle tattiche prestazionali è di generare una risposta a un evento in input al sistema con dei limiti temporali, controllandone il periodo in cui essa viene generata. Visti i vincoli imposti dalla GraphAPI di Facebook e il tempo di processamento delle chiamate REST ad essa è importante che in Cicero tali richieste vengano

ottimizzate. Inoltre anche la scelta del DBMS deve riflettere la natura dei dati da persistere. Salvare il grafo sociale dell'utente e della sua comunità nella rigida struttura tabellare di un approccio relazionale sarebbe altamente inefficiente.

Si consideri, ad esempio, lo scenario di un grafo i cui nodi rappresentino delle persone e gli archi la relazione di amicizia tra loro. Usando un RDBMS bisogna creare la tabella *FriendOf(A, B)* ed aggiungere due ennuple per ogni tripla. Si supponga che gli archi non siano orientati e che, per semplicità di rappresentazione, si intendano per A e B gli identificatori dei nodi tra cui sussiste la relazione di amicizia. Per navigare il grafo delle amicizie è necessario effettuare un *join* per ogni livello di distanza tra due nodi nel grafo.

Ne consegue un forte degrado delle prestazioni, che risulta proporzionale al numero di *hop* necessari, in quanto il *join*, seppur ottimizzato nei moderni sistemi, è l'operazione più costosa all'interno di un database relazionale. In algebra relazionale tale *join* potrebbe essere così descritto: $FriendOf_0 \bowtie_{B_0=A_1} FriendOf_1 \bowtie \ldots \bowtie_{B_(i-1)=A_i} FriendOf_i$, dove i è distanza tra il nodo iniziale e l'ultimo considerato.

Utilizzare al posto di un RDBMS un graph db consente una rappresentazione naturale della rete sociale di amicizie, la navigazione della quale è possibile tramite operazioni primitive. Per questo motivo è stato scelto Neo4j come strumento di persistenza dei dati estratti da Facebook.

5.2.3 Tattiche per la sicurezza

La sicurezza è una misura della capacità del sistema di assicurare l'accesso a persone e sistemi autorizzati e, nel contempo, proteggere i dati da chi non lo è. Varie tattiche architetturali possono essere impiegate per garantire ciò.

In Cicero la protezione della privacy è fondamentale, in quanto il sistema esamina informazioni strettamente personali di ogni utente. Pertanto l'accesso viene limitato ai soli utenti autenticati e che abbiano verificato il proprio account su Facebook. Ciò viene reso possibile tramite l'introduzione del protocollo di auto-

rizzazione *OAuth2.0*[2], che consente a un'applicazione di terze parti di ottenere un accesso limitato a un servizio HTTP, o per conto del proprietario della risorsa, tramite la gestione di un'interazione di permesso tra questi e il servizio HTTP, o consentendo all'applicazione di terze parti di ottenere l'accesso da sè.

Tramite OAuth2.0 in Cicero, di fatto, viene confermata l'identità dell'account Facebook dell'utente e controllate quali informazioni personali questi voglia condividere con il sistema. Inoltre gli utilizzatori del framework hanno accesso solo ai propri dati e non possono leggere altri profili.

In aggiunta, alle persone che desiderino rimuovere il proprio account su Cicero e non usufruire più delle sue raccomandazioni viene consentito di eliminare definitivamente dal database tutti i propri dati.

5.3 Pattern architetturali

Cicero presenta un'architettura *multi-tier*. Questi è un pattern architetturale della categoria di allocazione, in quanto mappa elementi software su elementi elaborativi. In un contesto distribuito spesso nasce il bisogno di separare l'infrastruttura in strutture, sia hardware che software, computazionalmente distinte e connesse tra loro tramite un determinato canale. Multi-tier si pone all'interno di tale scenario. L'idea su cui si fonda è organizzare le strutture d'esecuzione in gruppi logici di componenti. Ciascuno di questi insiemi è chiamato *tier*. La suddivisione può seguire vari criteri, come il tipo di componente, la condivisione di uno stesso ambiente di esecuzione o perchè svolgono la stessa funzione a runtime.

Punti di forza del pattern sono la suddivisione in moduli, ciascuno dei quali con le proprie responsabilità e possibilità di riuso, ottimizzazione delle prestazioni, facilità nel garantire la sicurezza. Anche la modificabilità viene incrementata: i sottogruppi, computazionalmente diversi, necessitano di un accordo sui protocolli con cui interagire, riducendo di fatto il mutuo accoppiamento.

[2]http://tools.ietf.org/html/rfc6749

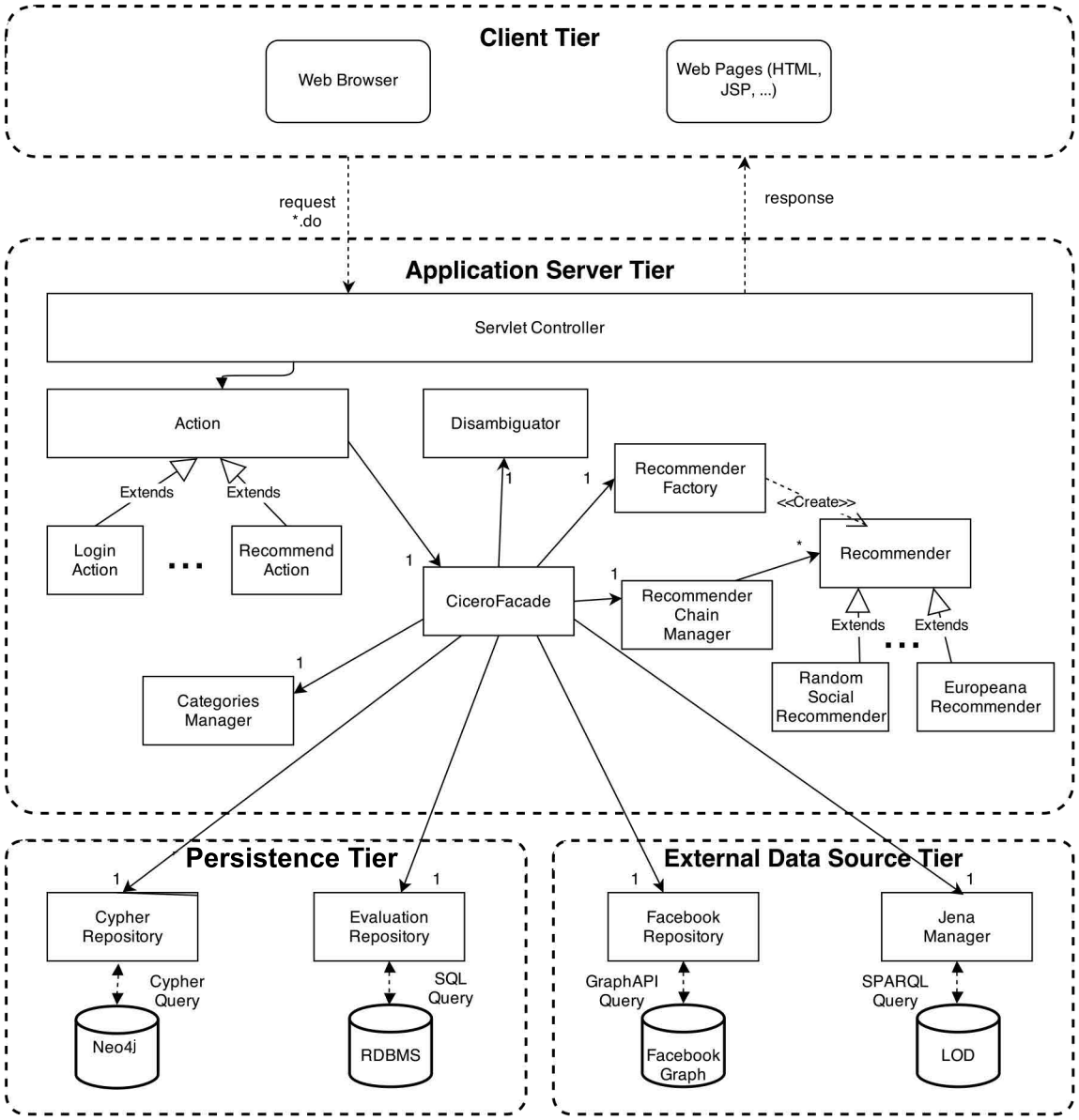

Figura 5.1: Vista logica dell'architettura software di Cicero

In Cicero sono stati individuati quattro tier principali:

- il tier del client (*Client Tier*);
- il tier della logica applicativa (*Application Server Tier*);
- il tier della persistenza (*Persistence Tier*);
- il tier che si occupa della connessione con sorgenti dati esterne (*External Data Source Tier*).

La Figura 5.1 illustra la vista logica dell'architettura. Due aspetti emblematici da sottolineare sono la suddivisione schematica in tier distinti ma in grado di interagire tra loro ed un approccio descrittivo prettamente orientato agli oggetti, simile a un diagramma delle classi: sono infatti evidenziati in essa i principali oggetti e le relazioni tra loro.

5.3.1 Client Tier

Il *Client Tier* raggruppa tutti quei componenti dell'architettura che si occupano dell'interazione con l'utente. Ciascun client è caratterizzato da un comune web browser sui cui sono fatte girare della pagine HTML e JSP per la presentazione dell'applicazione. Essi sono di fatto *thin*, nel senso che il compito ad essi attribuito è la sola interazione con l'utente. Questi è in grado di selezionare delle operazioni, sotto forma di richieste all'Application Server Tier, e di vederne visualizzate le risposte così generate. Le richieste sono nella forma di HTTP Request, mentre le risposte sotto forma di HTTP Response.

5.3.2 Application Server Tier

L'Application Server Tier è lo strato architetturale che ha la responsabilità di gestire la logica applicativa, coordinare il sistema ed effettuare decisioni. Riceve

le richieste inoltrate dal client, le elabora interagendo con gli altri tier e restituisce i risultati direttamente al client affinchè questi possa visualizzarli e renderli accessibili all'utente. Il server tier è centralizzato ed è fisicamente allocato in una macchina distinta dagli altri tier. Riesce a gestire efficacemente le richieste multiple provenienti da più client. Esso si avvale di *Apache Tomcat*[3], un application server che agisce da container di Servlet e Java Server Pages e fornisce un Web Server HTTP in grado di eseguire in modo nativo codice Java.

In questa sezione l'attenzione si concentra sui vari componenti architetturali del tier e le loro interazioni. Il fondamento logico e i motivi su cui tali scelte si fondano, soprattutto da un punto di vista dei design pattern, sono trattati in dettaglio nei successivi paragrafi. Nel seguito si descrivono le classi che compongono questo tier.

Application Server Tier espone i propri servizi ai client tramite un unico entry point, caratterizzato dalla classe *ServletController*. Essa implementa un controller: intercetta tutte le HTTP Request che soddisfano l'espressione regolare "\.do$" (quali *StoreFacebookFriend.do* oppure *Recommend.do*), stringhe che corrispondono a operazioni consentite dal sistema, le interpreta, instanzia ed esegue le relative *Action* (come *LoginAction* o *RecommendAction*), restituendone così il risultato al client come opportune pagine JSP. Varie implementazioni della classe Action sono state sviluppate. Generalmente ciascuna di queste azioni corrisponde a una o più operazioni dei vari casi d'uso. *LoginAction* permette all'utente di effettuare il login al proprio account di Facebook, in modo che il sistema possa autenticarlo e consentirgli di accedere ai servizi forniti da Cicero. *ClearGraphDatabaseAction* ha il compito di cancellare interamente il graph database di Neo4j associato all'utente corrente. Questa funzionalità è stata aggiunta per garantire la privacy. *RecommendAction* ha la responsabilità di raccomandare gli item sulla base degli algoritmi selezionati dall'utente e notificati a tale Action tramite opportuni parametri all'interno della HTTP Request. *StoreFacebookUserDataAction* salva

[3]http://tomcat.apache.org/

in modo permanente all'interno del graph database tutti i dati sociali relativi all'utente, come nome, cognome, ID di Facebook, amici, luoghi visitati e loro categorie.

StoreFacebookFriendDataAction analogamente al precedente, persiste in Neo4j le informazioni sociali degli amici dell'utente, soffermandosi in particolar modo sui posti in cui essi hanno assegnato un tag su Facebook e corrispondenti categorie.

Per garantire le funzionalità qui elencate, tali classi Action hanno necessariamente bisogno di interagire con gli altri moduli del server e del tier. L'oggetto *CiceroFacade* è una facade a cui le varie Action si rivolgono per tale scopo. Essa ha la responsabilità di comunicare, all'interno dell'Application Server Tier, con *Disambiguator, CategoriesManager, RecommenderFactory, RecommenderChainManager* e con gli altri tier tramite i rispettivi entry point.

Disambiguator espone metodi per la disambiguazione dei tag dei posti visitati dall'utente. *CategoriesManager* calcola, tra tutte le categorie assegnabili a una Facebook Page, quelle più simili agli interessi dell'utente. *RecommenderFactory* crea istanze di oggetti che implementano Recommender, in base all'algoritmo selezionato precedentemente dall'utente. *RecommenderChainManager* è il modulo che gestisce la pipeline di raccomandatori. Analogamente alla classe Action, varie implementazioni di raccomandatori sono state sviluppate: ognuna di queste classi corrisponde a una delle strategie descritte nel Capitolo 4. Si ha quindi la classe *RandomSocialRecommender*, *EuropeanaRecommender*, *DBpediaCloserPlacesRecommender* e via discorrendo.

5.3.3 Persistence Server Tier

Il *Persistence Server Tier*, costituito da server database, è lo strato dell'architettura che si occupa della persistenza dei dati, sia su database a grafi (Neo4j) che su quelli relazionali (PostgreSQL è il RDBMS adottato nello sviluppo di Cicero). I dati di questo livello sono mantenuti disaccoppiati da applicazioni server o da logica di business. Fornendo informazioni del proprio livello inoltre migliora

la scalabilità e le prestazioni. Espone i propri servizi tramite opportune classi, ovverosia *CypherRepository* per le operazioni che si interfacciano con Neo4j, *EvaluationRepository* per quelle che necessitano di RDBMS.

Come descritto nei precedenti capitoli, l'impiego di due database management system differenti nasce dalla necessità di avvalersi delle tecnologie più consone al proprio dominio e al problema che si deve affrontare. Laddove Neo4j si rivela particolamente efficiente ed elegante nel modellare e interrogare un grafo sociale, PostgreSQL e il mondo relazionale risultano la soluzione migliore per salvare i dati relativi alle valutazioni degli utenti. Infatti essi sono strutturati e facilmente rappresentabili come tabelle. CypherRepository ed EvaluationRepository forniscono entrambi le principali funzioni CRUD (Create, Read, Update, Delete), non solo nella forma base, ma anche in quelle più avanzate. Ad esempio, per il primo si possono memorizzare le relazioni di amicizia tra due utenti, il fatto che un posto sia stato visitato da più persone, appartenga a più categorie e abbia coordinate geografiche precise nel primo, mentre per il secondo si possono considerare i dati dell'utente come sesso, lavoro, istruzione, età e i voti attribuiti alla qualità dei raccomandatori testati. In particolare, CypherRepository si avvale del potente linguaggio di query *Cypher*, sviluppato dal team di Neo4j, con cui possono essere elaborate interrogazioni espressive, eleganti ed efficienti sul dominio a grafo, sfruttando il pattern matching. Cypher è inoltre dichiarativo: si focalizza, cioè, non su come estrarre i dati ma su cosa estrarre, in un modo facilmente comprensibile da un essere umano.

Invece EvaluationRepository interroga PostgreSQL, come un qualsiasi RDBMS, tramite statement SQL.

5.3.4 External Data Source Tier

L'*External Data Source Tier* ha la responsabilità di interfacciarsi con le sorgenti dati esterne al sistema, in particolare con le API di Facebook e con gli endpoint dei Linked Open Data.

Facebook offre agli sviluppatori la *Graph API*[4], giunta alla versione 2.2 al momento della stesura di questa tesi. Essa è il metodo principale per le applicazioni per leggere e scrivere i dati sul grafo sociale. Si presenta come una API a basso livello basata sul protocollo HTTP: permette, ad esempio, di elaborare query sui dati sociali, pubblicare nuove storie, caricare foto e video, e via discorrendo.

Un'applicazione Java come Cicero può usufruire dei servizi forniti dagli endpoint delle Graph API tramite chiamate GET o POST ed analizzarne le risposte, che saranno in formato JSON.

Di seguito vengono elencati alcuni esempi di richieste possibili, sfruttate di fatto anche in Cicero:

- *GET graph.facebook.com/v2.1/user-id/feed* consente di recuperare i post più recenti presenti sul diario dell'utente, inclusi gli aggiornamenti di stato e i link pubblicati da lui o da altri sul suo feed;
- *GET graph.facebook.com/v2.1/user-id/tagged* permette di visualizzare solo i post in cui la persona è stata taggata;
- *GET graph.facebook.com/v2.1/user-id/photos* restituisce le foto dell'utente.
- *GET graph.facebook.com/v2.1/user-id/friends* ritorna nome e ID Facebook degli amici dell'utente;
- *GET graph.facebook.com/v2.1/user-id/tagged_places* produce una risposta con tutti i posti (identificati univocamente da una Facebook Page corrispondente) in cui la persona ha un tag geolocalizzato.

La componente del sistema che si occupa delle interazioni con le Graph API è la classe *FacebookRepository*. Essa offre metodi che implementano le richieste GET precedentemente esposte e ne cattura le risposte inviate dagli endpoint, costruendo entità corrispondenti del modello, come un *CiceroPlace* o una *Category*.

[4]https://developers.facebook.com/docs/graph-api

Per quanto riguarda il Semantic Web, invece, il modulo dell'architettura del sistema che vi si interfaccia è *JenaManager*. Questi, tramite opportune query formulate in SPARQL, ha la responsabilità di esplorare la vastità di risorse espresse come LOD ed estrarne le triple RDF in modo che tali informazioni possano essere efficacemente elaborate dagli altri componenti.

Per gli scopi di Cicero, è stato creato un *mirror* di DBpedia su un server locale (ospitato dal Laboratorio di Intelligenza Artificiale dell'Università degli Studi Roma Tre, in cui questa tesi è stata sviluppata) e interrogato tale endpoint. In particolare, il dataset di DBpedia è stato reso persistente e interrogabile localmente tramite il triple store di Virtuoso Server. L'utilizzo di un endpoint SPARQL locale si è rivelato una soluzione valida, soprattutto in termini di prestazioni e di disponibilità. Ciò ha garantito così agli sperimentatori un servizio costantemente online durante tutto il loro processo di sperimentazione del sistema. Infatti l'endpoint pubblico di DBpedia[5], essendo usato massivamente in tutto il mondo, presenta numerosi periodi di sovraccarico e limita perciò l'utilizzo al raggiungere di un prefissato numero di richieste ai propri servizi. Siccome sono numerose le interrogazioni che Cicero effettua sul LOD DBpedia, ad esempio per le operazioni di disambiguazione o per quelle di raccomandazione, lunghi tempi di mancata disponibilità del servizio non sono tollerabili. Per quanto riguarda Europeana, invece, il problema non sussiste essendo inferiore il numero di richieste che il suo endpoint SPARQL[6] deve gestire. Pertanto, non è stato necessario installare un suo mirror in locale.

5.4 Design pattern adottati

In Ingegneria del Software i design pattern sono delle soluzioni generalmente riusabili a un problema che occorre frequentemente all'interno di un determinato contesto nella progettazione di componenti software. Essi sono fondamentali ed

[5] http://dbpedia.org/sparql
[6] http://europeana.ontotext.com/sparql

è presente una vastissima trattazione in letteratura, in particolare in [EG04] e in [Lar04]. Non sono dei design finiti che possono essere trasformati direttamente in codice ma delle descrizioni o linee guida su come affrontare un problema in differenti situazioni. Sono quindi delle *best pratice* formalizzate a cui i programmatori possono ricorrere nella fase di design di un'applicazione o di un sistema.

I design pattern orientati agli oggetti sono soliti mostrare relazioni e interazioni tra classi o oggetti, senza specificare in dettaglio il risultato finale.

All'interno del framework proposto è stato fatto un ampio uso dei design pattern: nelle prossime sottosezioni si riporta una loro descrizione e i motivi per cui sono stati adottati.

5.4.1 Singleton

Il pattern creazionale *Singleton* assicura che una classe abbia un'unica istanza, ovvero un "singleton", fornendo un punto globale e singolo per accedere ad essa. Un classe che segue questo approccio è essa stessa l'unica responsabile per tenere traccia della sua sola istanza e assicurarsi che non ve ne siano altre in esecuzione.

Solitamente si ottiene ciò definendo un metodo statico della classe che restituisca il singleton. In Cicero è stato invece adottato *Google Guice* [7], un framework Java di dependency injection sviluppato da Google che, tra i vari vantaggi, annovera anche quello di favorire la gestione delle classi Singleton in modo snello ed elegante.

Infatti, sono varie le classi del framework che necessitano la presenza di un'unica istanza durante l'esecuzione del programma: ad esempio *ServletController*, *RecommenderFactory*, *RecommenderChainManager*, *CiceroFacade*, *JenaManager*, *FacebookRepository*, *CypherRepository*, *EvaluationRepository*.

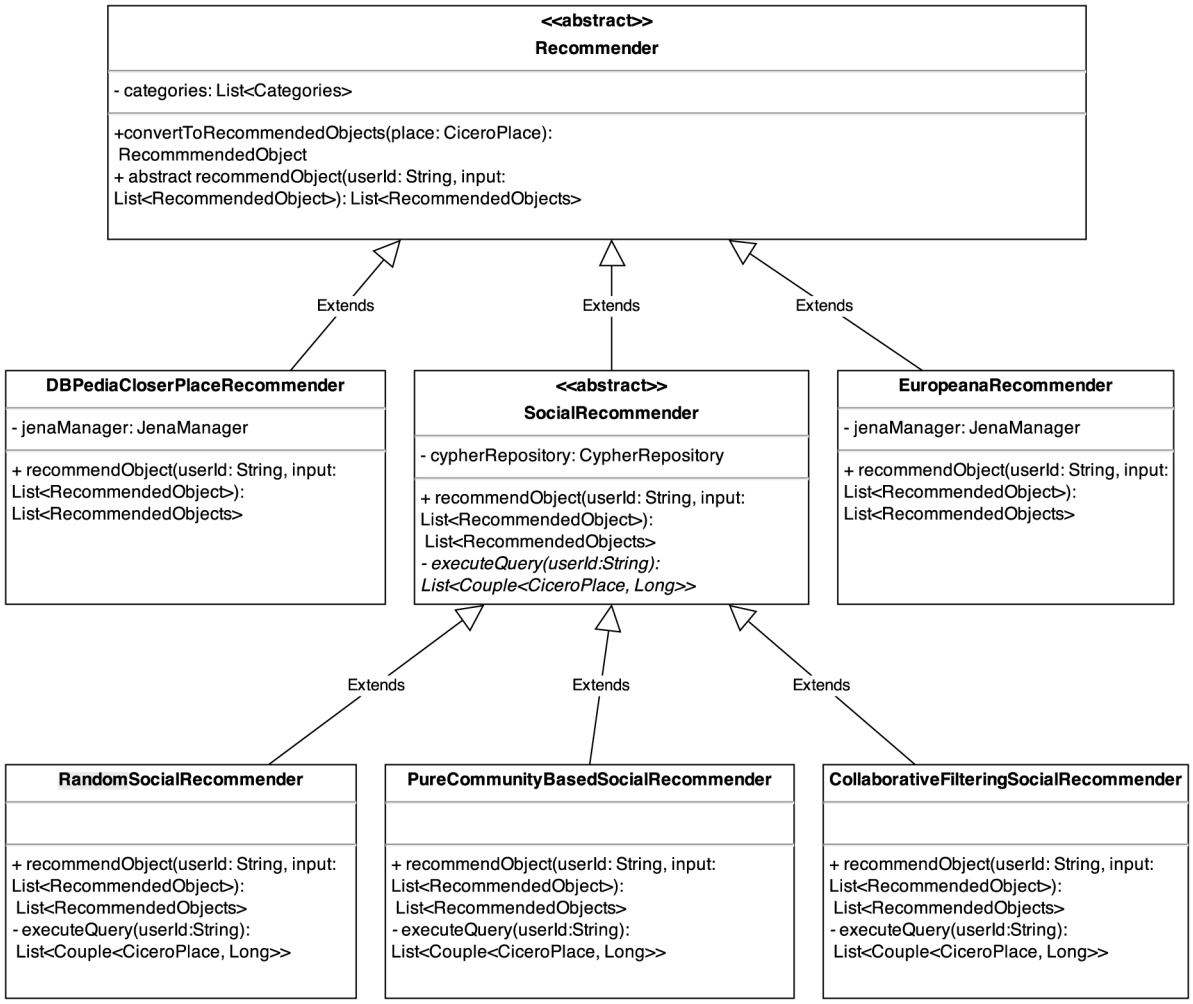

Figura 5.2: Il pattern Template Method applicato alla classe astratta Recommender e alle sue sottoclassi

5.4.2 Template Method

Il contesto in cui il pattern comportamentale *Template Method* viene applicato è quello della presenza di un algoritmo o di un comportamento con un core stabile ma con diversi punti di variazione. Il problema è che si devono implementare e mantenere più porzioni di codice abbastanza simili tra loro.

La soluzione fornitaci da questo pattern è quella di introdurre una famiglia di classi polimorfe. Permette di definire la struttura, il template di un algoritmo, come interfaccia o superclasse astratta, lasciando alle sottoclassi il compito di implementarne il comportamento che varia, sovrascrivendo le porzioni di codice opportune. In questo modo si evita la duplicazione del codice: l'ossatura del flusso di lavoro è implementata una sola volta nella superclasse e i cambiamenti necessari sono sviluppati nelle sottoclassi. La modificabilità viene favorita: nuove strategie possono essere aggiunte sovrascrivendo semplicemente gli specifici dettagli del flusso di lavoro.

Il Template method viene applicato in Cicero nella definizione dei Raccomandatori. Il RS implementato, infatti, si avvale di numerose varianti di algoritmi di raccomandazione. Pertanto, deve essere garantita da un lato una loro facile implementazione e, dall'altro, la possibilità di introdurre i nuovi sistemi di raccomandazione che in futuro verranno proposti in letteratura, senza dover apportare cambiamenti radicali all'architettura.

La Figura 5.2 illustra le modalità in cui tale pattern è stato applicato in questo contesto. La classe astratta *Recommender* contiene il metodo `RecommendedObject convertToRecommendedObjects ( CiceroPlace place )` che viene ereditato dalle sue sottoclassi e il metodo astratto `List<RecommendedObject> recommendObjects( String userId, List<RecommendedObject> input)` che viene da queste sovrascritto per implementare logiche di raccomandazioni differenti. Per aggiungere, quindi, un nuovo algoritmo di raccomandazione, è sufficiente crea-

[7] https://github.com/google/guice

re una nuova classe che estenda Recommender ed effettui un *override* del metodo RecommendObjects.

Classi concrete di Recommender sviluppate all'interno di Cicero sono *EuropeanaRecommender*, *DBpediaCloserPlacesRecommender* per quanto riguarda le tecniche di raccomandazione che sfruttano i Linked Open Data e *SocialRecommender* che analizza i dati sociali. Quest'ultima è, a sua volta, un tipo astratto. Essa, infatti, implementa un'ossatura comune dell'algoritmo all'interno di recommendObjects e lascia alle sue sottoclassi il compito di estendere la procedura `List<Couple<CiceroPlace, Long» executeQuery(String userId)`, per estrarre le informazioni dal grafo sociale tramite comportamenti variabili. Le tre classi concrete che estendono SocialRecommender sono *PureCommunityBasedSocialRecommender*, *RandomSocialRecommender* e *CollaborativeFilteringSocialRecommender*. Per approfondimenti su tali tecniche si rimanda al Capitolo 4 di questa tesi.

5.4.3 Factory Method

Un problema che occorre frequentemente è l'assegnazione della responsabilità di creazione di oggetti in presenza di condizioni particolari, ad esempio di una famiglia di classi poliforme che implementano un'interfaccia o che estendono una classe astratta: in questo contesto spesso non si sa al momento della compilazione quale particolare sottoclasse concreta verrà istanziata a run-time.

La soluzione viene fornita dal pattern *Factory Method*. Esso prevede la definizione di un oggetto "Factory" a cui attribuire il ruolo di *Creator*: questi definisce un'interfaccia per la creazione di oggetti polimorfi, lasciando tuttavia la selezione del tipo alle sue sottoclassi, creazione che verrà eseguita a runtime e ritornerà una qualsiasi implementazione dell'interfaccia.

I vantaggi introdotti da questo design pattern sono molteplici:

- si favorisce la modificabiltà, in quanto l'aggiunta o la sostituzione di una

classe concreta è immediata e trasparente al client;

- si separano fra loro le responsabilità della creazione di oggetti di supporto coesi;
- viene nascosta la logica di creazione potenzialmente complessa;
- si possono introdurre strategie per la gestione della memoria con cui migliorare le prestazioni, tramite il *caching* o il *pooling* delle istanze ad esempio.

Relativamente a Cicero è presente la famiglia di classi polimorfe che estendono l'interfaccia *Recommender*. La loro creazione è delegata alla factory singleton *RecommenderFactory* che, a tempo di esecuzione, instanzia la corretta classe concreta che rispetti le necessità dei casi d'uso. Infatti può capitare che l'utente voglia testare le raccomandazioni di Cicero usando una prima volta il DBpedia Closer Places Recommender e una seconda l'Europeana Recommender. Ciò viene garantito in modo naturale dall'adozione del pattern Factory Method.

5.4.4 Repository

Il pattern *Repository* è proposto per la prima volta in [Eva04] e qui descritto come una rappresentazione di tutti gli oggetti di un certo tipo nella forma di un insieme concettuale, che agisce come una collezione avanzata in grado di elaborare query complesse. E' una soluzione comunemente adottata quando occorre gestire lo strato di persistenza. Le classi Repository offrono metodi per eseguire le operazioni CRUD, non solo quelle di base, ma anche quelle più avanzate. Si configurano così come unico punto di accesso alle strutture di persistenza sottostanti, quali database relazionali, nonrelazionali, file XML e via discorrendo. Il costo è l'aggiunta di un livello ulteriore di indirezione al sistema. Tuttavia viene favorita la modularità del codice, i principi GRASP di alta coesione e basso accoppiamento, la modificabiltà. Si immagini lo scenario in cui, durante iterazioni future

dello sviluppo del framework, cambino i requisiti di sistema e si renda necessario il passaggio da una tecnologia di storage a un'altra (da Neo4j a un database document-oriented quale MongoDB, ad esempio). Sarebbe sufficiente sostituire i metodi che, all'interno del Repository, garantiscono le operazioni CRUD, con altri idonei al nuovo DBMS adottato. Tutto ciò risulterebbe assolutamente trasparente ai componenti dell'architettura.

In Cicero sono state sviluppate diverse classi Repository. *CypherRepository* si interfaccia con i graph database di Neo4j degli utenti, elaborando query avanzate tramite il linguaggio Cypher. Essa assicura che possano essere inseriti nuovi nodi, quali utenti, punti di interesse, categorie, locazioni geografiche e le relazioni tra di esse, come il fatto che due persone siano amiche tra loro all'interno del social network o che un individuo abbia già visitato un determinato luogo, evitando così che le informazioni salvate siano ridondanti o duplicate. Permette che un dato salvato possa essere recuperato in maniera efficiente, tramite non solo query primitive ma anche più elaborate: a tal proposito è possibile, ad esempio, effettuare collaborative filtering sugli item con una singola interrogaazione, restituendo i posti visitati dagli utenti con gli stessi propri interessi. Rende inoltre possibile la cancellazione totale del grafo.

EvaluationRepository interroga i database relazionali, in particolare PostgreSQL, per coprire tutte quelle funzionalità CRUD inerenti al salvataggio dei rating attribuiti dagli utenti ai vari raccomandatori, dopo averli testati e aver risposto alle domande del questionario di valutazione del sistema.

FacebookRepository si connette, per mezzo della libreria *RestFB*[8] o di chiamate REST, agli endpoint delle Graph API di Facebook. In questo modo è in grado di estrarre le informazioni relative al grafo sociale del profilo Facebook dell'utente. Gestisce inoltre i token d'accesso dell'applicazione.

JenaManager svolge il ruolo di Repository per i dataset del Web Semantico, diventando l'entry point del framework ad essi. Interroga gli endpoint dei progetti

[8]http://http://restfb.com/

DBpedia ed Europeana con opportune query SPARQL: l'obiettivo è quello di ricavare valori semantici e item aggiuntivi con cui espandere il contenuto del grafo sociale. Analogamente a FacebookRepository, offre per lo più metodi di lettura. Ad esempio, a partire da un dato punto di interesse, recupera da Europeana tutte quelle risorse collegate semanticamente ad esso, per titolo, soggetto o tipo, oppure da DBpedia tutti quei luoghi che risultano essere fisicamente vicini ad esso e, nel contempo, condividono una stessa categoria semantica.

Capitolo 6

Sperimentazione

In questo capitolo sono presentate le varie prove sperimentali cui è stato sottoposto il sistema di raccomandazione oggetto del lavoro di tesi.

In particolare, sono state valutate le varie tecniche adottate utilizzando diverse metriche, fra le quali *l'accuratezza percepita* degli item consigliati, la *novità*, la *serendipità* e la *diversità*. Essendo la qualità della raccomandazione un fattore altamente soggettivo, si è scelto di valutare la bontà dei vari algoritmi impiegando una scala di Likert a cinque punti. I vari rating collezionati tramite un questionario sono analizzati secondo le metriche NDCG e e sottoposti ad ANOVA Test per valutare la loro significatività statistica.

6.1 Dataset

La sperimentazione del sistema Cicero è stata effettuata su dataset reali, raccolti dallo stream delle attività degli utenti sul social network Facebook. L'adozione di dati reali si configura come un valore aggiunto rispetto all'analisi su dataset artificiali. Si evitano, infatti, spiacevoli incongruenze tra i dati che minerebbero l'accuratezza e la genuinità dei risultati.

E' stato chiesto a un campione di utenti con un account attivo sul social network Facebook di partecipare al test su base volontaria. Vista la sensibilità dei

dati da trattare, la tutela della privacy per questi utenti assume una priorità assoluta ed è garantita come segue. In primo luogo l'applicazione propone loro, al primo login su Facebook tramite le pagine di Cicero, la scelta di quali permessi assegnarle. In particolare, viene richiesta la possibilità di leggere i seguenti attributi:

- informazioni personali, quali nome, cognome, età, sesso, luogo di nascita, istruzione, professione;
- i post pubblicati dall'utente e gli aggiornamenti di stato;
- i luoghi in cui è stato eseguito un checkin o assegnato un tag;
- foto e album di foto;
- lista degli amici;
- feed degli amici, corredati con foto e tag geolocalizzati;

In secondo luogo, un caso d'uso del sistema prevede l'eliminazione definitiva del grafo sociale dell'utente dal database Neo4j di Cicero, non appena concluse le operazioni di sperimentazione.

Alla valutazione del framework ha partecipato un campione di 25 individui, la cui composizione demografica è schematizzata nella Tabella 6.1. Per migliorare l'usabilità del framework ed evitare di richiedere allo sperimentatore ulteriori passi di configurazione del sistema allo sperimentatore, i dati demografici sono stati raccolti in modo implicito e trasparente. Infatti, le informazioni circa il sesso, l'eta, la professione, la città di residenza, l'educazione possono essere facilmente recuperate dagli account di Facebook tramite opportune interrogazioni del sistema alla Graph API. Ciò è possibile, sempre ai fini del rispetto della privacy, solo se l'utente ha inserito in precedenza questi dati nel proprio profilo e se ha attribuito i permessi di lettura a Cicero.

I numeri relativi ai post e alle foto analizzate durante la fase di sperimentazione sono riportati nella Tabella 6.2. In media, ognuno degli sperimentatori ha una comunità di amici composta da 447 individui. La prima colonna della tabella mostra le statistiche di un utente singolo, la seconda dell'utente combinato con la sua cerchia di amici (sostanzialmente i valori della prima colonna moltiplicati per 448, il numero medio di amici a cui si somma 1, cioè l'utente stesso). La terza, infine, considera il dataset globale, includendo il campione intero dei 25 sperimentatori. In totale sono stati esaminati 5.967.450 post e 2.179.125 immagini. Dalla seconda colonna si nota come, dei 238.698 post e delle 87.165 foto estratte dallo stream delle attività della comunità di amici a cui un account sul social network appartiene, solo il 5,24% e 22,56 %, rispettivamente, ha un tag geolocalizzato annesso. Di conseguenza, gli algoritmi di filtering collaborativo proposti in Cicero hanno a disposizione, in media, per profilo, un campione di 32.184 posti visitati (di cui 12.516 provengono dagli status dell'utente e della sua cerchia di amici e 19.668 dalle foto) con cui generare una modellazione sociale per l'utilizzatore del sistema. Da questi numeri merge, inoltre, un comportamento sociale secondario che concerne l'aggiunta di un tag geolocalizzato alle proprie risorse, funzionalità propria dei LBSN: infatti le statistiche relative al campione analizzato evidenziano come gli utenti preferiscano maggiormente condividere la propria posizione geografica sul social network quando pubblicano foto (61,1%), rispetto a quando scrivono nuovi post sul proprio status (38,9%).

6.2 Valutazioni system-oriented vs user-centric

Come proposto in letteratura, (e.g., in [CGT13]) la qualità di un sistema di raccomandazione può essere stabilita secondo due famiglie di metriche opposte: *system-oriented* o *user-centric*. Nel primo approccio, il sistema di raccomandazione è valutato sulla base di dataset precostruiti di item afferenti a un dato dominio. Gli utenti non interagiscono con il sistema durante lo svolgimento del

Tabella 6.1: Composizione demografica del campione degli sperimentatori.

	Classe	Quantità	Percentuale
Sesso	M	13	52%
	F	12	48%
Età	<20	3	12%
	20-24	18	72%
	25-30	2	8%
	>30	2	8%
Educazione	Diploma di Maturità	3	12%
	Laurea Triennale	20	80%
	Laurea Magistrale	2	8%
Professione	Studente	23	92%
	Lavoratore	2	8%

Tabella 6.2: Statistiche relative al numero di post e foto analizzate durante la fase di sperimentazione del sistema.

	Utente	Utente e suoi amici	Dataset Totale
Numero medio Post Pubblicati	534	238.698	5.967.450
Numero medio Foto Pubblicate	195	87.165	2.179.125
Numero medio Post con posizione	28	12.516	312.900
Numero medio Foto con posizione	44	19.668	491.700
Numero Medio Amici per Utente	447		
Numero Utenti Sperimentatori	25		

test: la valutazione, in termini di accuratezza, è effettuata confrontando le opinioni sugli item stimate dal sistema di raccomandazione tramite i propri algoritmi e i giudizi precedentemente collezionati da utenti reali sugli stessi oggetti.

I metodi user-centric, come si evince dal nome, riguardano il punto di vista dell'utente: questi interagisce con un sistema di raccomandazione in esecuzione e riceve da esso consigli di item in real time. Le misure sono collezionate interrogando direttamente l'utente, tramite interviste o questionari, osservando il suo comportamento durante l'uso e memorizzando in modo automatico le sue interazioni (e.g., i click effettuati o le pagine visualizzate). Tra i due approcci cambiano

anche le metriche da utilizzare: mentre tra i system-oriented si analizzano metriche oggettive come *precision*, *recall* o *fallout*, al pari di un classico sistema di Information Retrieval o Machine Learning, negli user-centric vengono introdotti nuovi parametri, come la *serendipità* o la *diversità*, che riguardano maggiormente la soggettività e la percezione del sistema da parte dell'utente, non calcolabili facilmente con formule o algoritmi. In Cicero si è scelto di adottare una soluzione user-centric per la sperimentazione, ricorrendo allo strumento *questionario*.

6.3 Questionario di valutazione del sistema

Il questionario costituisce un valido strumento per stimare le qualità di un sistema di raccomandazione secondo una logica user-centric. Esso consiste nel porre delle domande mirate all'utente che ha testato il prodotto, per comprendere il risultato della sua esperienza e quanto questi sia effettivamente soddisfatto. Quando si gestiscono i questionari, si assume ovviamente che l'algoritmo di raccomandazione da valutare operi online. Il questionario sviluppato nella sperimentazione di Cicero prende spunto dalle riflessioni contenute all'interno del lavoro documentato in [PCH11]. Esso è composto da varie domande, ognuna delle quali mira ad analizzare un parametro saliente.

L'*accuratezza percepita (perceived accuracy)* esprime quanto gli utenti considerino la raccomandazione vicina ai propri interessi e preferenze. In altri termini, rappresenta una stima generale di quanto il sistema abbia "indovinato" i gusti dell'utente. Come dimostrato in [CP09], ad una risposta positiva a questa domanda consegue che sia altamente probabile che l'algoritmo sottostante sia effettivamente accurato nel predire gli interessi dell'utente.

La *diversità (diversity)* misura quanto gli oggetti siano percepiti come dissimili tra loro nella lista degli item consigliati: una persona non è soddisfatta, infatti, se gli oggetti suggeriti da un Recommender System sono tutti simili tra loro e non variano nel corso del tempo.

La *novità (novelty)* calcola fino a che punto gli utenti ritengono di ricevere raccomandazioni nuove e interessanti. Il suo concetto base è legato alla capacità del recommender di suggerire all'utente item di cui egli ignorava l'essitenza. E' un parametro molto importante per la percezione che l'utente ha del sistema nel suo complesso.

La *serendipità (serendipity)* è una metrica strettamente correlata alla novità. Non solo l'utente ha conosciuto un nuovo item, ma riconosce che difficilmente avrebbe effettuato tale scoperta percorrendo altre vie.

6.3.1 Modalità di sperimentazione

La valutazione del sistema si compone di alcuni passaggi. In una prima fase Cicero deve acquisire sufficienti dati per modellare l'utente e comprenderne gli interessi. Dopo aver avuto accesso al proprio account di Facebook tramite l'interfaccia di Cicero, gli sperimentatori consentono al framework di estrarre e rendere persistenti prima le proprie informazioni sociali, (quali nome, città di residenza, sesso, lavoro, istruzione, punti di interesse visitati) e, successivamente, i dati relativi ai propri amici, in particolare tutti quei luoghi taggati su un post o una foto.

La seconda fase è quella della sperimentazione vera e propria. Cicero propone all'utente una lista di item raccomandati attraverso una particolare pipeline di algoritmi, senza che questi ne conosca i dettagli. Insieme a tale lista, allo sperimentatore viene sottoposto un questionario con alcune domande. Le risposte ad esso sono chiuse e seguono la cosiddetta scala di Likert a 5 punti, presentata nell'articolo [Edm05].

Questa tipologia di scale psicometriche è stata introdotta dal Dr. Rensis Likert, sociologo presso l'Università del Michigan, con lo scopo di assegnare "scientificità" alle misure dei comportamenti psicologici. Agli intervistati viene chiesto di indicare il loro livello di accordo/disaccordo con ciascuna delle affermazioni proposte, scegliendo una tra cinque (o sette) possibili risposte: completamente

d'accordo, d'accordo, né d'accordo né in disaccordo, disaccordo, completamente in disaccordo. A ciascuna risposta viene attribuito un punteggio che varia da 5 (completamente d'accordo) ad 1 (completamente in disaccordo).

Ad ognuno degli item suggeriti dall'algoritmo, lo sperimentatore deve assegnare un voto che rispecchi il livello di corrispondenza con i suoi reali gusti e interessi. Tali rating contribuiscono a definire il grado di accuratezza percepita.

Dopo aver segnalato al sistema il feedback su ciascun item, l'utente, infine, deve indicare il grado di accordo/disaccordo a tre ulteriori affermazioni. Esse sono relative alla lista dei risultati considerata nel suo complesso, con lo scopo di esaminare le proprietà di novità, serendipità e diversità, come riportato in Figura 6.1.

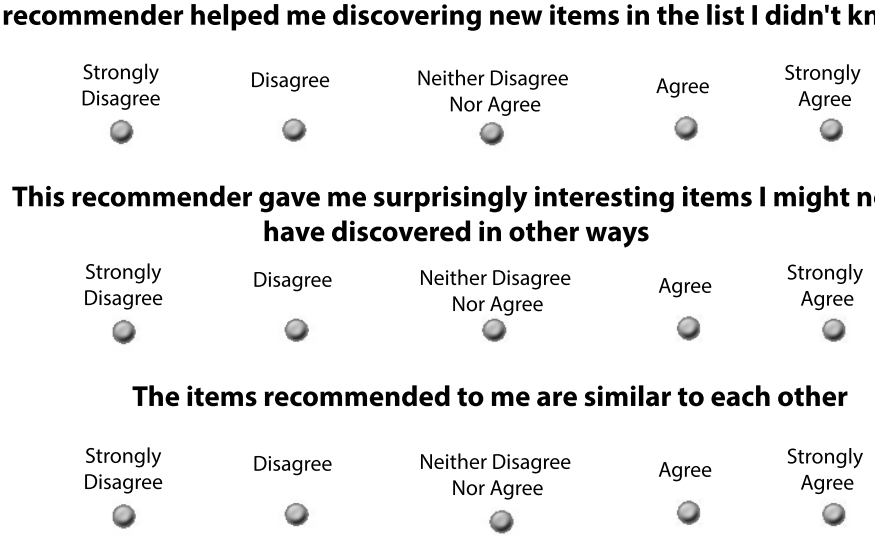

Figura 6.1: Affermazioni a cui gli sperimentatori devono esprimere il proprio grado di accordo/disaccordo mediante una scala di Likert a 5 punti. Esse sono contenute all'interno del questionario di valutazione del sistema proposto agli sperimentatori dopo che essi hanno ricevuto la lista degli item raccomandati da una sequenza di algoritmi. Tali affermazioni sono relative, rispettivamente, alle proprietà di novità, serendipità e diversità.

Nello specifico le affermazioni sono:

- "Il raccomandatore mi ha aiutato a scoprire nuovi elementi nella lista che non conoscevo prima" per la novità;
- "Il raccomandatore mi ha fatto scoprire elementi inaspettatamente interessanti in cui non mi sarei potuto imbattere percorrendo altre vie" per la serendipità;
- "Gli item che mi sono stati raccomandati sono simili tra loro" per la diversità.

Tali affermazioni, usando la scala di Likert a 5 punti, presuppongono che la pipeline testata suggerisca all'utente una lista di item non vuota. In realtà, potrebbe capitare che, per mancanza di informazioni sufficienti, si verifichi il problema del cold-start ed il sistema non sia in grado di fornire alcuna raccomandazione all'utente, restituendo un result set nullo. Durante la fase di analisi dei voti, si può notare l'occorrenza di tale situazione: infatti essa corrisponde a quando lo sperimentatore non ha fornito alcuna risposta al questionario a lui sottoposto. A tal proposito, i risultati in Cicero sono molto soddisfacenti, in quanto, per il campione di utenti considerato, non sono stati registrati casi di risposte mancanti. Da ciò consegue che il RS è stato sempre in grado di contrastare il cold-start.

La fase di valutazione viene ripetuta nove volte (per testare tutte le possibili sequenze di algoritmi di raccomandazione previste nella versione attuale del sistema Cicero), proponendo di volta in volta all'utente una lista prodotta da un raccomandatore differente e il rispettivo questionario da compilare.

6.4 Analisi e discussione dei risultati sperimentali

6.4.1 Valutazione in termini di NDCG

Varie tecniche sono state proposte in letteratura per analizzare i risultati di una sperimentazione e comprendere se, e quanto, sia positivo il sistema di raccomandazione sviluppato [MRS08]. Fra le più diffuse, anche in ambito di Information Retrieval (soprattutto nelle ricerche web e i relativi task) è la *Discounted Cumulative Gain* (DCG), e la sua versione normalizzata, *Normalized Discounted Cumulative Gain* (NDCG), proposte entrambe in [JK02]. Essa misura l'utilità (rilevanza) di un documento restituito in base alla posizione che esso occupa nella lista degli item. Tale metrica si basa su due assunzioni fondamentali:

- i documenti molto rilevanti sono più utili di quelli rilevanti solo marginalmente;
- più è bassa la posizione in classifica di un documento rilevante, più è difficile che esso venga esaminato, rendendolo così meno interessante per l'utente.

NDCG presenta vantaggi importanti rispetto ad altre metriche di valutazione [WWL+13]. In primo luogo consente a ciascun item recuperato di avere un rating associato alla rilevanza, mentre le misure tradizionali prevedono unicamente una rilevanza binaria: ciascun documento è visto come pertinente o non pertinente.

In secondo luogo, NDCG si avvale di una funzione di riduzione (discount) sul rank, al contrario delle altre tecniche che assegnano i pesi in modo uniforme alle diverse posizioni. Questa caratteristica è particolarmente importante nei motori di ricerca, dal momento che agli utenti interessano soprattutto i documenti in cima alla classifica.

Per calcolare il valore di NDCG di una lista di documenti restituiti bisogna procedere per passi. Innanzitutto è necessario calcolare il Discounted Cumulative

Gain. Questi è basato sul *Cumulative Gain* (CG), in cui il guadagno è accumulato partendo dal primo elemento restituito fino a quello che si trova in posizione p

$$CG@p = \sum_{i}^{p} rel_i \tag{6.1}$$

dove rel_i è la rilevanza graduata (gain) dell'item in posizione i. Il DCG è calcolato riducendo (da cui il termine discounted) il CG nei rank inferiori. Tale sconto è direttamente proporzionale alla posizione del documento nella lista suggerita. Un tipico sconto è $\frac{1}{log(rank)}$. Se la base del logaritmo è 2, il discount al rank 4 vale $\frac{1}{2}$, al rank 8, $\frac{1}{3}$, con una riduzione abbastanza regolare. Difatti la probabilità che un utente esamini un documento diminuisce in funzione del suo rank, in base alla formula:

$$DCG@p = \sum_{i}^{p} \frac{rel_i}{log_2(i)} \tag{6.2}$$

NDCG normalizza il valore di DCG con il *DCG Ideale* (IDCG), ovvero il DCG che avrebbe la lista di recommended item perfetta e più inerente agli interessi dell'utente.

$$NDCG@p = \frac{DCG@p}{IDCG@p} \tag{6.3}$$

dove con la notazione NDCG@p si indica il valore di NDCG al rank p, che è particolamente adottato nella sperimentazione di motori di ricerca. Nello specifico vengono analizzati NDCG@1, NDCG@5 e NDCG@10. Rank maggiori di 10 sono solitamente di scarso interesse scientifico e non vengono pertanto considerati. Il valore di NDCG è sempre compreso tra 0 e 1. Il ranking perfetto per ogni query si ottiene ordinando i documenti in ordine decrescente di rel_i e calcolandone il DCG. Infatti il ranking ideale ritornerebbe prima gli item con il livello più alto di rilevanza, successivamente quelli con il valore inferiore di rilevanza e via discorrendo.

In questo paragrafo vengono presentati e confrontati tra loro i valori medi di NDCG ottenuti dagli algoritmi proposti in Cicero durante la fase di sperimentazione. L'obiettivo principale è quello di valutarne l'*accuratezza*. Per ogni lista di

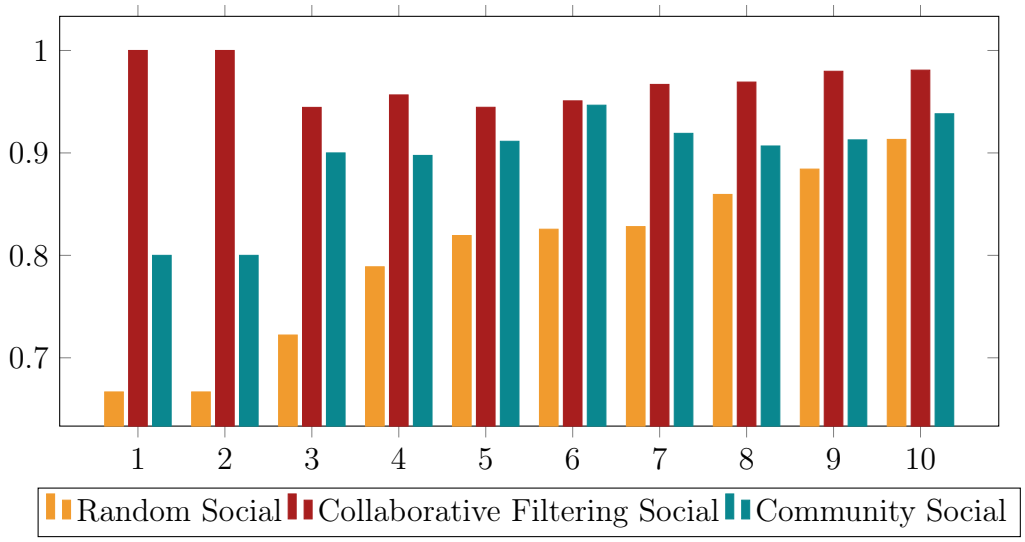

Figura 6.2: NDCG Medio Totale relativo al Gruppo 1, composto dagli algoritmi *Random Social*, *Collaborative Filtering Social* e *Community Social*, ciascuno rappresentato da un colore univoco, come mostrato dalla legenda annessa. Si tratta di un risultato sul dataset globale, al contrario delle prossime due figure, dove i voti raccolti sono distinti in base alla nazione di residenza dello sperimentatore. L'asse delle ascisse indica il rank p, mentre quello delle ordinate il corrispondente valore di *NDCG*.

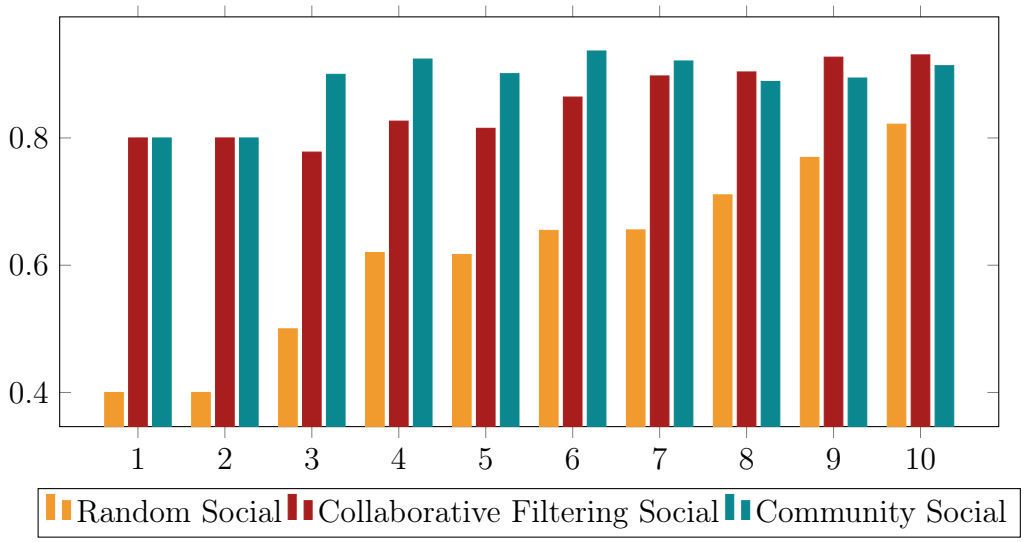

Figura 6.3: NDCG Medio Italia, Gruppo 1.

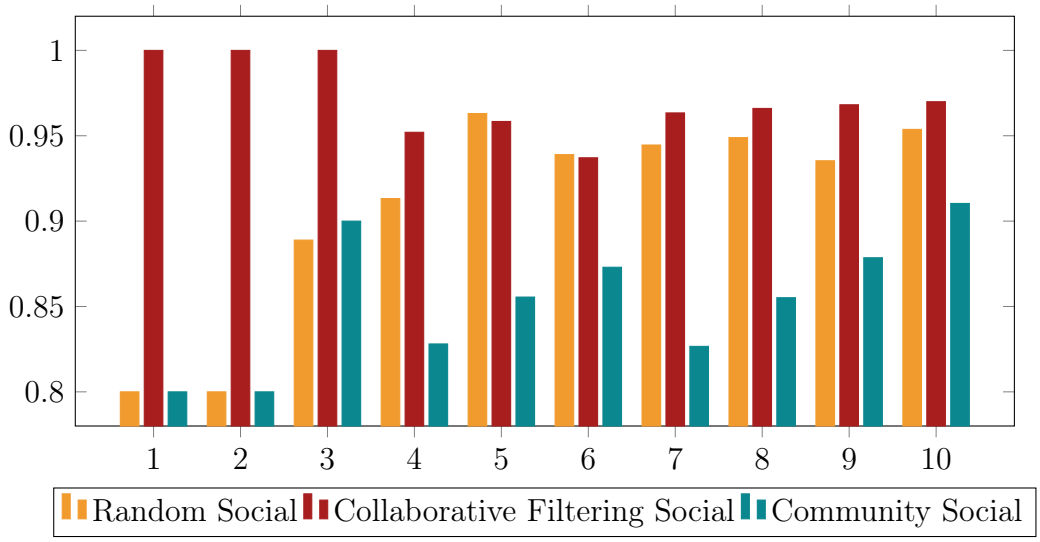

Figura 6.4: NDCG Medio Europa, Gruppo 1.

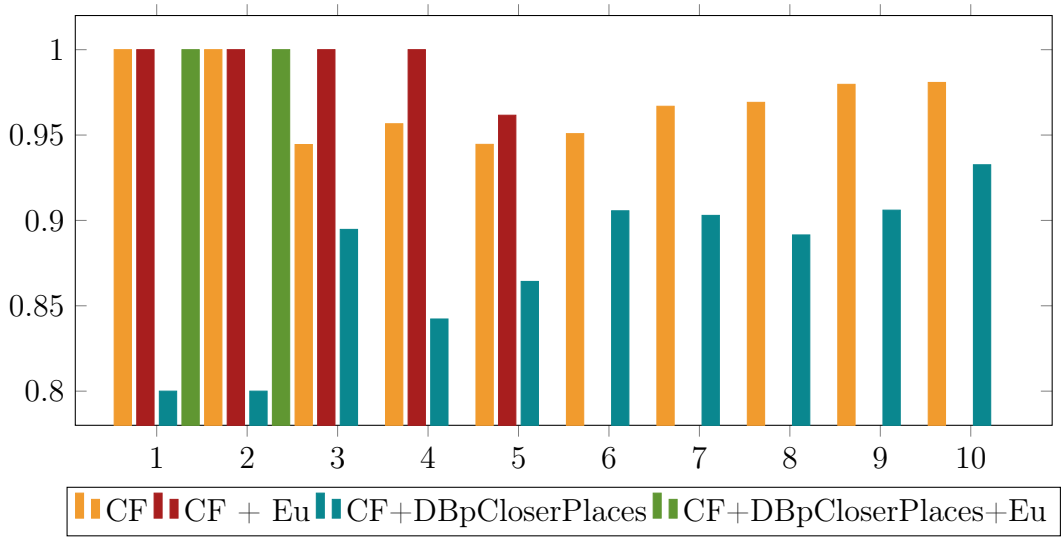

Figura 6.5: NDCG Medio Totale relativo al Gruppo 2, composto dalle seguenti pipeline di raccomandatori: *Collaborative Filtering (CF), CF + Europeana Recommender (Eu), CF + DBpediaCloserPlaces, CF + DBpediaCloserPlaces + Eu.*

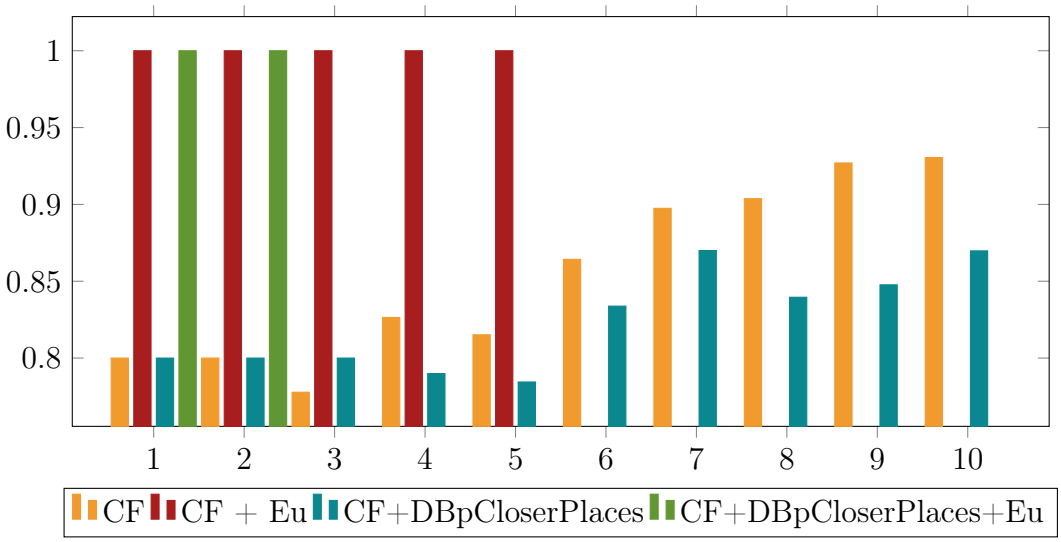

Figura 6.6: NDCG Medio Italia, Gruppo 2.

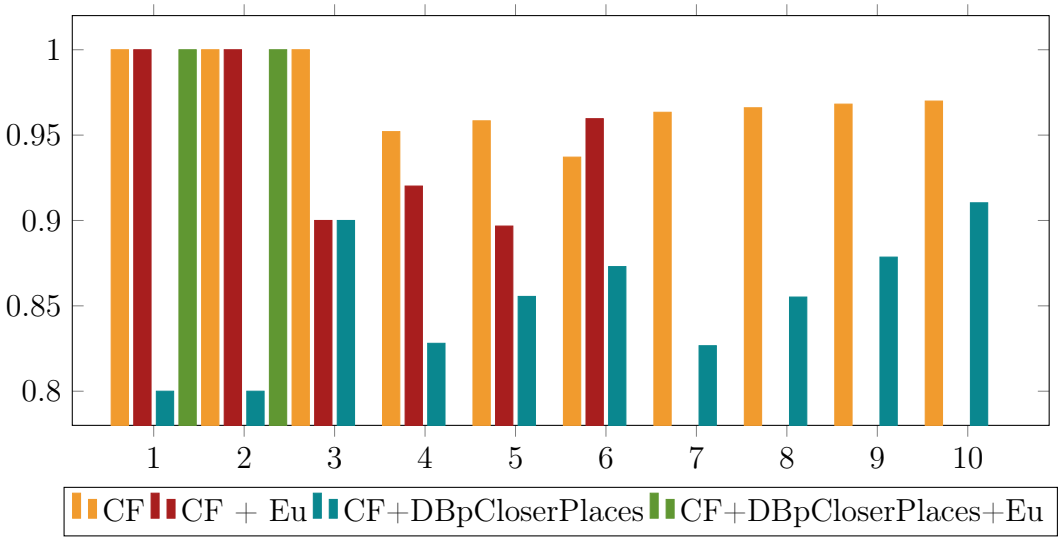

Figura 6.7: NDCG Medio Europa, Gruppo 2.

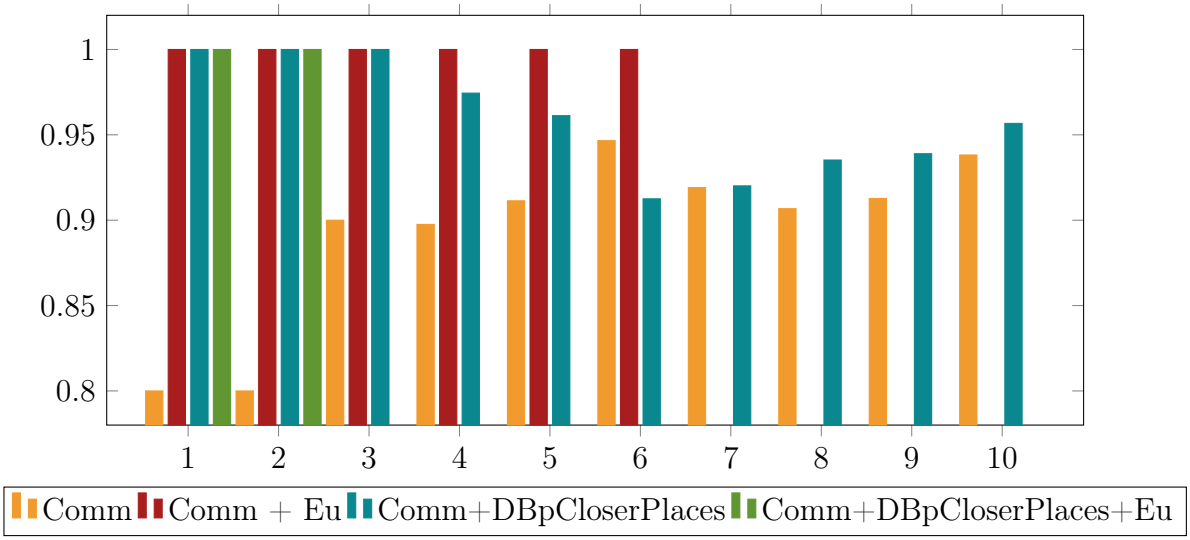

Figura 6.8: NDCG Medio Totale relativo Gruppo 3, costituito dalle seguenti sequenze di algoritmi di raccomandazione: *Community-based (Comm), Comm + Eu, Comm + DBpediaCloserPlaces, Comm + DBpediaCloserPlaces + Eu.*

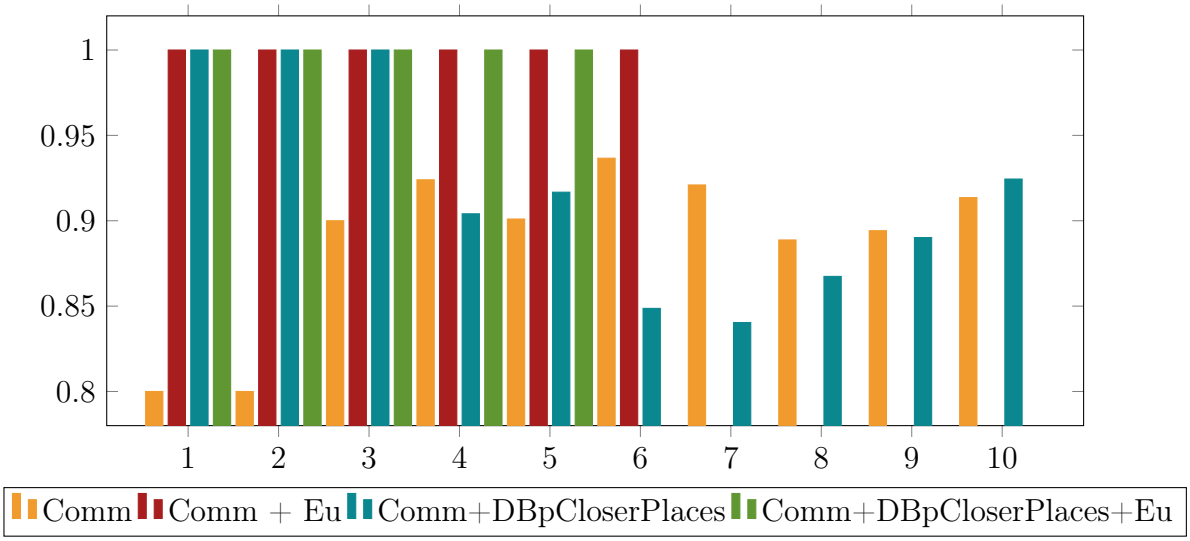

Figura 6.9: NDCG Medio Italia, Gruppo 3.

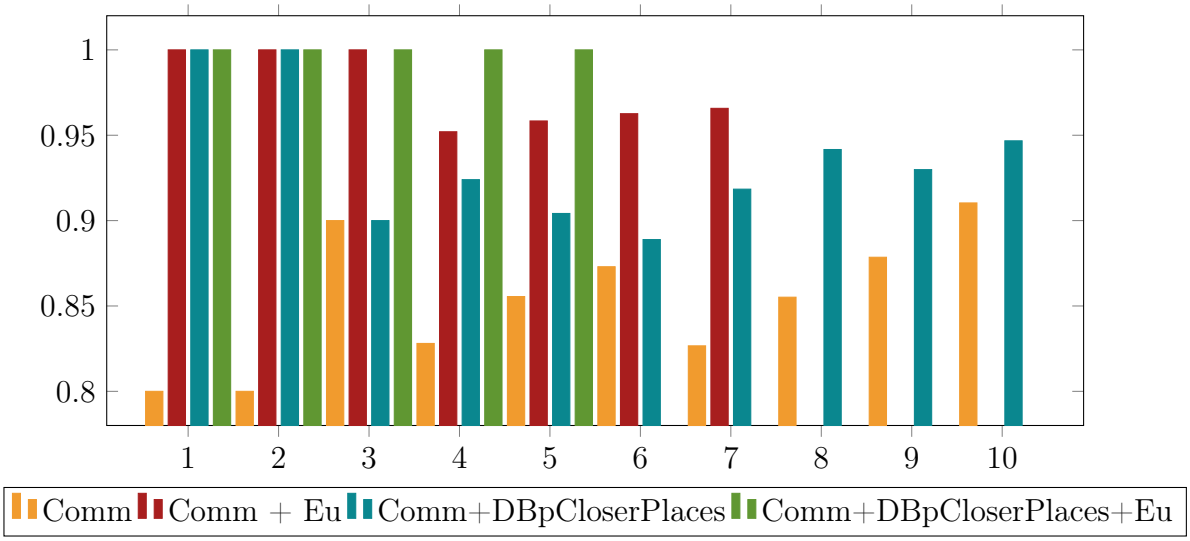

Figura 6.10: NDCG Medio Europa, Gruppo 3.

oggetti restituiti dal sistema di raccomandazione scelto, viene chiesto all'utente di assegnare un voto a ciascuno degli item, in base al grado di pertinenza con i suoi interessi. Il rating è espresso nel range di una scala di Likert a 5 punti, come illustrato in precedenza, e convertiti i valori da 1 a 5: 5 se l'elemento è ritenuto altamente rilevante dall'utente, 1 se, al contrario, è completamente irrilevante. In particolare, delle valutazioni collezionate dagli utenti per ogni algoritmo, viene calcolato il voto medio per ogni rank. Tali medie rappresentano il gain rel_i di ciascun documento e sono usate così per ricavare il risultato di NDCG@p.

Sono state effettuate varie sequenze di test. Innanzitutto si è ritenuto opportuno aggregare i dati non tanto per sesso, per età o per professione, quanto per luogo di residenza. Infatti Cicero crea modelli utente a partire da un Location-Based Social Network: l'89% dei tag geolocalizzati effettuati dagli sperimentatori, estratti dalle loro attività sul social network, hanno coordinate geografiche interne della propria città di residenza. Inoltre, gli stessi item raccomandati sono dei punti di interesse geografici, come il Colosseo, il Museo di Van Gogh ad Amsterdam o un fiordo norvegese, soprattutto quando sono coinvolti nella pipeline di raccomandazione algoritmi sociali o basati su DBpedia. Per cui risulta interes-

sante analizzare il comportamento del framework in presenza di dati appartenenti ad aree geografiche distinte, visto che il grafo sociale inferito e le risultanti raccomandazioni sono altamente influenzate da esse. La composizione del campione

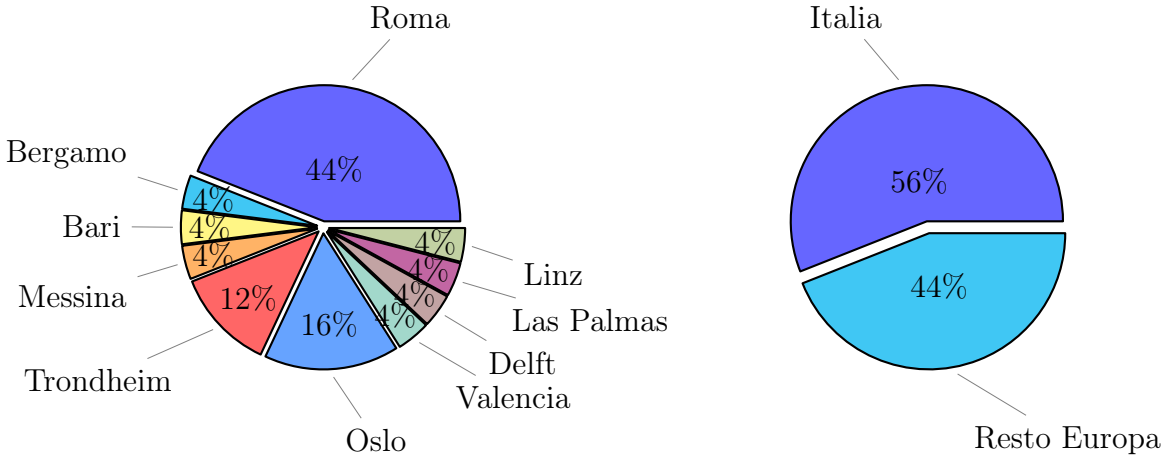

Figura 6.11: Dati degli sperimentatori in relazione al loro luogo di residenza. Nel grafico di sinistra i campioni sono aggregati per città di residenza; in quello di destra sono suddivisi per nazione di appartenenza, in particolare Italia o resto d'Europa. Queste informazioni sono inferite estraendo, in maniera implicita, il campo "hometown" dei profili facebook degli sperimentatori.

sperimentale è illustrata in Figura 6.11. In particolare, nei test effettuati si è tenuta in considerazione l'aggregazione per nazione: da una parte il gruppo che accomuna gli utenti provenienti da città italiane (56%), dall'altra quello con gli individui che vivono in altre nazioni europee (44%). Questa è la soluzione ritenuta migliore per ottenere una sperimentazione più accurata, non essendoci, città di Roma a parte, alte percentuali di tester appartenti alla stessa città o nazione.

I test effettuati considerano quindi tre scenari: Italia, Resto d'Europa e intero dataset. Per ciascuno di essi vengono confrontati tra loro i risultati in termini di NDCG ottenuti da tre insiemi di pipeline di algoritmi:

- Il primo gruppo è composto dagli algoritmi sociali puri, senza ulteriori integrazioni: *Random Social Recommender, Collaborative Filtering Social Recommender* e *Community-Based Social Recommender*;

- Il secondo è formato da tutte le possibili pipeline che coinvolgono il *Collaborative Filtering Social Recommender (CF)*. Include quindi *CF* singolo, *CF + Europena Recommender, CF + DBpedia Closer Places Recommender, CF + DBpedia Closer Places + Europeana Recommender*;

- Infine il terzo include le pipeline in cui è presente il *Community-Based Social Recommender (Comm)*. E' composto dal *Comm* singolo, *Comm + Europena Recommender, Comm + DBpedia Closer Places Recommender, Comm+DBpedia Closer Places + Europeana Recommender*.

Nel corso della sperimentazione sono state effettuate nove combinazioni di test (illustrate nelle Figure dalla 6.2 alla 6.10). L'asse delle ascisse indica il rank p, mentre l'asse delle ordinate il corrispondente valore di *NDCG@p*, con $NDCG@p \in [0, 1]$. Ogni pipeline è rappresentata, in ciascun istogramma, con un colore differente, la cui corrispondenza è riportata nella legenda ad esso annessa.

La prima serie di test mira a mostrare come, nell'ambito degli algoritmi che analizzano l'aspetto sociale dell'utente e ne profilano gli interessi, una tecnica di raccomandazione randomica non sia efficace. L'accuratezza, infatti, delle metodologie basate sull'approccio community-based e sul filtering è di gran lunga superiore, come testimonia la Figura 6.2. Da essa si evince come, per ogni rank, il Random Social Recommender abbia un valore di NDCG inferiore rispetto alle altre due versioni, soprattutto nei rank bassi.

Nelle Figure da 6.5 a 6.10, si vuole invece confrontare l'accuratezza delle pipeline che includono o meno strumenti semantici nelle loro raccomandazione. Da tali istogrammi emerge come le versioni che presentano una componente semantica consentano di ottenere prestazioni migliori in termini di NDCG rispetto alla controparte che sviluppa solo l'aspetto sociale. Il contributo della semantica influenza soprattutto il valore di NDCG nei rank medi, prossimi al 5, quando l'algoritmo sociale coinvolto è il Collaborative Filtering (Figura 6.5). Quando invece è presente la tecnica sociale fondata sull'approccio collaborativo, l'apporto fornito

dall'arricchimento del profilo utente con il Linked Open Data è decisamente più netto e favorevole. Come risulta anche nell'istogramma di Figura 6.8, l'NDCG delle pipeline composte è maggiore di quella contenente il solo Community-Based Recommender e assume valore 1 in diversi posizioni del rank. Quando l'NDCG raggiunge l'unità, o comunque è molto prossimo ad essa, significa che l'algoritmo è stato in grado di comprendere appieno gli interessi dell'utente e di comportarsi come un raccomdandatore ideale. Un valore di NDCG uguale ad 1 non si ottiene unicamente in presenza di voti massimi (quindi 5, in questo caso). E' sufficiente, infatti, che l'ordine degli item restituiti sia decrescente in termini di rilevanza. Come argomentato nel paragrafo precedente, la metrica NDCG mira a valutare se, e quanto, i documenti rilevanti siano presenti nelle posizioni superiori della lista. Nel caso delle tecniche qui proposte, la media dei voti risulta prossima ai 4,2.

Un'altra considerazione che si può trarre analizzando tali grafici è il fatto che le prestazioni dei sistemi di raccomandazione siano positive nei risultati ottenuti dalla sperimentazione su entrambe le categorie (Italia e Resto d'Europa). Nello scenario italiano le prestazioni sono complessivamente migliori, probabilmente grazie alla maggiore omogenità degli utenti in termini di luoghi visitati. Cicero, quindi, agisce in modo soddisfacente nel fornire raccomandazioni di POI a una qualsiasi tipo di utente, a prescindere da quale sia la nazione o la città di residenza.

Inoltre, i dari ottenuti evidenziano una sostanziale indipendenza delle prestazioni dalla particolare sorgente LOD introdotta nel processo di raccomandazione. In alcuni casi l'NDCG di un algoritmo che adotta Europeana è migliore, come in Figura 6.6, con una differenza media del 20%. In altre situazioni, invece, DBpedia consente di ottenere suggerimenti più accurati, come in Figura 6.10. Tuttavia l'azione combinata di Europeana e DBpedia porta sempre ai maggiori valori di NDCG nei vari scenari.

Un aspetto da sottolineare soprattutto per le tipologie di pipeline *CF + Europeana + DBpedia Closer Places e Comm + Europeana +DBpedia Closer Places*,

è che i documenti restituiti non vanno oltre il rank 5. Da un lato, l'intersezione dei due dataset restringe inevitabilmente il numero totale di item da suggerire (ad esempio, non è detto che una risorsa culturale consigliata con l'impiego di DBpedia abbia una corrispondente rappresentazione come foto, video, testo, audio nell'archivio di Europeana). Dall'altro, i pochi item restituiti hanno tuttavia una notevole rilevanza per l'utente, così da far registrare NDCG di valore unitario. Si può concludere, quindi, che l'impiego di Europeana in un sistema di raccomandazione di POI e di beni artistici e culturali è assolutamente consigliato.

Inoltre, tra le due pipeline complete, quella con un approccio sociale community-based consente di ottenere un numero maggiore di item rispetto alla controparte con il collaborative filtering, una media di 5 item nel result set del primo contro i 2 del secondo. Il fenomeno è anche da addebitare alla differente natura delle due tecniche, in quanto quella collaborativa restringe notevolmente il dominio dei documenti, restituendo solo quelli apprezzati dagli utenti più simili; il community-based, d'altro canto, considera allo stesso livello tutti gli individui della comunità, consentendo al sistema di disporre di un insieme di selezione degli item da raccomandare molto più ampio.

6.4.2 Test di verifica della significatività statistica

Una sperimentazione reale completa prevede la verifica della significatività statistica dei risultati. In altri termini, occorre dimostrare che i valori osservati siano reali e non dovuti al caso. Ciò si traduce, nello scenario di Cicero, nelle verifica che gli algoritmi collaborativi siano effettivamente migliori di quello randomico e che i sistemi di raccomandazione che sfruttano le informazioni contestuali fornite dal Linked Open Data forniscano raccomandazioni più accurate delle controparti costituite dalla sola analisi della sfera sociale dell'utente.

Dei vari test proposti in letteratura è stato scelto l'ANalysis Of VAriance, o ANOVA [Fis25]. E' una tecnica appartenente al ramo della statistica inferenziale usata per determinare se le differenze nelle misure osservate delle variabili

dipendenti tra i gruppi siano dovute, o meno, a dei trattamenti differenti delle variabili indipendenti o a fluttuazioni casuali. L'obiettivo è quindi quello di valutare il livello di significatività di un esperimento. Nello specifico, con il termine *significatività* si intende la probabilità p con cui lo sperimentatore è favorevole a rigettare l'*ipotesi nulla* quando è in realtà corretta. Le soglie adottate generalmente sono 0.05 ($p = 5\%$, significativo) o 0.01 ($p = 1\%$, molto significativo). Tuttavia, il primo dei due valori è in genere sufficientemente piccolo da poter concludere che sia improbabile che la differenza osservata sia dovuta al semplice caso.

Quando ci si accinge a confrontare due o più gruppi di dati, l'ipotesi nulla prevede che non esista alcuna differenza tra questi insiemi riguardo al parametro considerato e le eventuali differenze osservate siano da imputare soltanto al caso. Nell'ANOVA Test viene calcolato il rapporto, detto *treatment index*, fra la variabilità esterna (*between*) e quella interna ad essi (*within*).

$$treatmentindex = \frac{Varianza\_Between}{Varianza\_Within} \tag{6.4}$$

La prima componente considera la variazione dei campioni trattati tutti nello stesso modo ed è legata perciò a caratteristiche comuni tra i gruppi. La seconda, invece, implica che i campioni siano assegnati in modo casuale a soltanto una delle condizioni di trattamento e rappresenta la differenza tra i vari trattamenti. Quindi la variabilità *between* è un fenomeno legato a caratteristiche proprie di ciascun algoritmo come il sociale o la semantica, nel caso di Cicero. In altri termini, se la varianza tra gli algoritmi è di gran lunga maggiore rispetto a quella interna ad un singolo algoritmo, è bassa la probabilità che la differenza tra questi algoritmi sia soltanto legata alla varianza interna.

Se il valore di treatment index ottenuto è vicino ad 1, si può ritenere che non vi siano effetti introdotti dalle caratteristiche proprie dell'algoritmo e che, pertanto, l'ipotesi nulla sia vera. Al contrario, quando questi risulta maggiore dell'unità, potrebbero essersi verificati delle conseguenze attribuibili ad esse. In questo caso,

Tabella 6.3: Risultati dell'ANOVA test sul primo insieme di pipeline. Sono presenti due tabelle: quella superiore riporta, per ogni gruppo, la media dei valori di NDCG@p ottenuti per i rank da 1 a 10 e la relativa varianza; quella inferiore, invece, racchiude altre informazioni significative, proprie dell'ANOVA, suddivise per tipo di varianza (*Between*, *Within* e totale): somma dei quadrati (*SQ*), gradi di libertà (*GDL*), media dei quadrati (*MQ*), *F-Ratio*, *significatività* e *F-Critico*.

Gruppi	**Media**	**Varianza**
Random Social	0.797413	0.007434
Community Social	0.893226	0.002653
Collaborative Filtering Social	0.969292	

Tipo varianza	**SQ**	**GDL**	**MQ**	**F-Ratio**	**Significatività**	**F-Crit**
Between	0.148362	2	0.074181	21.16133	2.96 E-06	3.354131
Within		27	0.003505			
Totale	0.24301	29				

tuttavia, per poter rigettare con certezza l'ipotesi nulla, bisogna assumere che sia vera un'*ipotesi alternativa*. Il treatment index, detto anche *F-Ratio*, può essere ricavato con la procedura statistica denominata *F-Test*. Solo nella situazione in cui il value di F ottenuto sia uguale o ecceda il valore critico, si può rigettare l'ipotesi nulla. Tale *F-Critico* si ottiene tramite opportune tabelle di criticità che considerano la probabilità p, il numero di gruppi e il numero di partecipanti.
Per verificare la significatività statistica dei risultati ottenuti a seguito degli esperimenti di accuratezza sugli algoritmi di raccomandazione, è stato effettuata un'analisi della varianza ad un fattore sui valori di NDCG ottenuti. Nello specifico, sono stati effettuati quattro ANOVA test: i primi tre confrontano i risultati ricavati da ciascuno dei tre gruppi di pipeline presentati in precedenza; il quarto test, invece, analizza la varianza di tutte le nove pipeline in un unico insieme. In tutti e quattro i casi sperimentati è stato possibile rigettare l'ipotesi nulla, in quanto il corrispettivo *F-Ratio* è risultato di gran lunga superiore al valore di *F-Critico*.

Le Tabelle 6.3, 6.4, 6.5 illustrano l'analisi della varianza dei tre scenari di pipeline presi singolarmente. La Tabella 6.6, invece, mostra i risultati del-

Tabella 6.4: Risultati dell'ANOVA test sul secondo insieme di pipeline.

Gruppi	**Media**	**Varianza**
CF	0.969292	0.000429
CF + DBpedia	0.873992	0.002121
CF + DBpedia + Europeana	1	0
CF + Europeana	0.992322	0.000295

Tipo varianza	**SQ**	**GDL**	**MQ**	**F-Ratio**	**Significatività**	**F-Crit**
Between	0.07303	3	0.024343	23.20295	3.83 E-07	3.354131
Within	0.02413	26	0.001049			
Totale	0.097161	26				

Tabella 6.5: Risultati dell'ANOVA test sul terzo insieme di pipeline.

Gruppi	**Media**	**Varianza**
Comm	0.893226	0.002653
Comm + DBpedia	0.95992	0.001102
Comm + DBpedia + Europeana	1	0
Comm + Europeana	1	

Tipo varianza	**SQ**	**GDL**	**MQ**	**F-Ratio**	**Significatività**	**F-Crit**
Between	0.060503	3	0.020168	16.11086	3.35 E-06	2.960351
Within	0.033799	27	0.001252			
Totale	0.094302	30				

l'ANOVA test che considera, come gruppi, ciascuna delle pipeline proposte dal sistema Cicero. Ciascuna tabella è composta da due sezioni. In quella superiore è presente il riepilogo dei dati da esaminare, nello specifico il *gruppo*, la *media* e la *varianza* dei valori di NDCG su tutti i rank (da 0 a 10, qualora presente). Quella inferiore, invece, descrive i valori del test di ANOVA veri e propri, quindi varianza tra gruppi (*between*), varianza interna al gruppo (within), *somma dei quadrati (SQ)*, *gradi di libertà (GDL)*, *media dei quadrati (MQ)*, *F-Ratio*, *significatività* e *F-Critico*. In particolare con *SQ*, si intende la somma degli scarti quadratici delle medie dei singoli gruppi dalla media totale, mentre con *MQ*, la somma degli

Tabella 6.6: Risultati dell'ANOVA test su tutte le pipeline proposte.

Gruppi	**Media**	**Varianza**
Random	0.797413	0.007434
Comm	0.893226	0.002653
Comm + DBpedia	0.95992	0.001102
Comm + DBpedia + Europeana	1	0
Comm + Europeana	1	0
CF	0.969292	0.000429
CF + DBpedia	0.873992	0.002121
CF + DBpedia + Europeana	1	0
CF+ Europeana	0.992322	0.000295

Tipo varianza	**SQ**	**GDL**	**MQ**	**F-Ratio**	**Significatività**	**F-Crit**
Between	0.326497	8	0.040812	19.28871	7 E-14	2.960351
Within	0.124835	59	0.0002116			
Totale	0.451332	67				

scarti quadratici dei singoli voti rispetto alla media del gruppo cui appartengono. *SQ* ed *MQ* vengono calcolate sia per la componente intragruppo che per quella intergruppo e sommate per ottenere il valore totale rispettivo.

GDL rappresenta invece i gradi di libertà. In ambito statistico, il *GDL* di una variabile aleatoria esprime la quantità minima di dati sufficienti per valutare la percentuale di informazione contenuta in una statistica.

6.4.3 Valutazione di novità, serendipità e diversità

Dopo aver riportato nei paragrafi precedenti l'*accuratezza percepita* degli algoritmi sociali e semantici proposti in Cicero, in questa sezione è esaminata la bontà del sistema dal punto di vista delle proprietà di *novità*, *serendipità* e *diversità* percepite. Tramite il questionario cui sono stati sottoposti gli utenti, sono stati collezionati i voti degli sperimentatori in merito a tale metriche. La Figura 6.12 illustra la media di tali rating sulle varie pipeline proposte. Da tali valori è

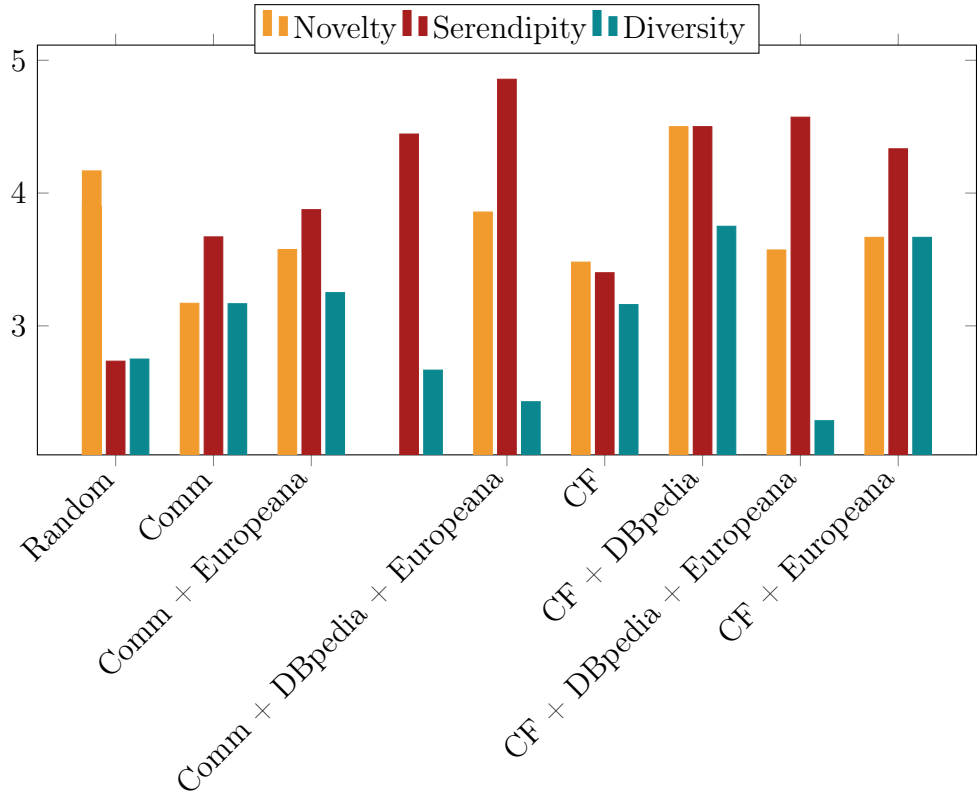

Figura 6.12: Media dei voti attributi dagli sperimentatori alle varie pipeline in merito alle metriche di novità, serendipità e diversità.

possibile trarre alcune conclusioni.

Innanzitutto il raccomandatore Random Social è quello che ha ottenuto valori di serendipità più bassi e risulta mediamente il peggiore tra gli algoritmi che si avvalgono della sola componente sociale. Quindi un approccio collaborativo o basato sulla comunità sono di gran lunga da preferirsi in un sistema di raccomandazione sociale. Risultato degno di nota è il punteggio medio positivo ottenuto nella metrica di novità, ovverosia 3.9/5. Questo fenomeno è da accreditare al fatto che, scegliendo casualmente item da raccomandare, è molto probabile che all'utente venga suggerita un'opera d'arte che non conosceva prima. Malgrado ciò, dal corrispondente basso livello di serendipità (2.74/5), si evince come l'utente non ritenga indispensabile l'aiuto del sistema di raccomandazione per scoprire tali

nuovi item, ma che avrebbe potuto raggiungere lo stesso obiettivo percorrendo altre strade.

Confrontando poi i gruppi che includono solo il fattore sociale con le versioni che sfruttano anche informazioni semantiche, emerge nuovamente come l'avvalersi della base di conoscenza del LOD porti a risultati migliori non solo nei valori dell'accuratezza percepita, come dimostrato nei paragrafi precedenti, ma anche dal punto di vista delle altre metriche. L'impiego combinato di DBpedia ed Europena apporta i valori più alti di serendipità (4.84/5), anche se gli approcci che si avvalgono di una sola delle due componenti consentono di ottenere prestazioni ugualmente positive. La metrica in cui si registrano i valori più bassi è la diversità, fra la quale il punteggio più alto è stato ottenuto da *CF+DBpedia* (3.75/5). In generale, gli utenti ritengono che gli algoritmi propongano loro liste contenenti item spesso simili tra loro.

Analogamente a quanto fatto per i risultati di test effettuati sull'accuratezza percepita, anche quelli relativi alle altre metriche sono stati sottoposti a verifica della loro significatività statistica.

In tutti i casi, essendo risultato il *treatment index* maggiore del valore critico per $p = 0.05\%$, è possibile rigettare l'ipotesi nulla e concludere che gli algoritmi proposti in Cicero hanno effettuato un impatto significativo sulle variabili soggettive di novità, serendipità e diversità.

Capitolo 7

Conclusioni e sviluppi futuri

In questa tesi si è illustrato Cicero, un sistema di raccomandazione sociale di beni appartenti al patrimonio artistico e culturale mondiale. Esso si avvale dell'analisi dei dati sociali degli utenti, ricavati dai loro account su Facebook e arricchiti con informazioni provenienti dai dataset Europeana e DBpedia, progetti cardine nel dominio dei Linked Open Data.

Nello specifico, il sistema ideato e sviluppato monitora le attività svolte dall'utente su importanti servizi del Social Web come Facebook, ne inferisce la semantica e fornisce strategie per collezionare le informazioni contestuali dal Web of Data, con lo scopo di generare profili utente semanticamente validi.

Il framework è stato presentato e sperimentato nel contesto dei sistemi di raccomandazione sociale geospaziali, dove il nocciolo della sfida risiede nel dedurre le preferenze utente per i punti di interesse relativi ai beni del patrimonio artistico e culturale, come dipinti, monumenti, musei e altre attrazioni turistiche. Per raggiungere tale scopo è stato, inoltre, presentato un algoritmo per la disambiguazione dei tag geolocalizzati dei post e delle foto degli account di Facebook. Esso si avvale di strumenti di *Named Entity Recognition* allo Stato dell'Arte (*TagMe* e *GeoNames*) e di un accurato confronto con le effettive coordinate geografiche del tag.

Le *research question* a cui si è voluto rispondere mediante lo sviluppo di Cicero

sono sintetizzabili nelle seguenti:

- Quanto influisce la scelta della sorgente dei dati sociali sulla modellazione degli interessi dell'utente e sulla qualità degli item raccomandati nel dominio dei beni appartenenti al patrimonio artistico e culturale?
- In che misura l'arricchimento semantico del profilo sociale di un utente, ottenuto interrogando i dataset pubblici aderenti alla Linked Open Data Cloud, come DBpedia ed Europeana, può fornire un contributo positivo al processo di raccomandazione sociale?
- Quali sono le più efficaci combinazioni di strategie di raccomandazioni in termini di accuratezza percepita, serendipità, novità e diversità?

Alla sperimentazione del sistema ha partecipato un campione composto da 25 volontari con un account attivo su Facebook. In particolare, il totale delle risorse sociali processate è stato di 5.967.450 post e 2.179.125 foto. Le risultanze scientifiche riscontrate a valle della valutazione di Cicero, in termini di NDCG e ANOVA Test per le proprietà di *accuratezza percepita*, *novità*, *serendipità* e *diversità*, sono state più che incoraggianti. Tuttavia, sono varie le possibili strade che si possono percorrere per migliorare ulteriormente l'attuale versione del sistema. Tali sviluppi traggono spunto da alcune considerazioni.

La prima è che il mondo dei Recommender System è in continuo mutamento. Dal punto di vista della profilazione dell'utente, soprattutto per quel che concerne i dati sociali estratti dagli account attivi sui social network, nuovi algoritmi di filtraggio di item organizzati seconda una topologia a grafo saranno verosimilmente proposti dalla comunità scientifica. Tali algoritmi potranno essere facilmente sviluppati in Cicero, grazie all'adozione del design pattern *Template Method*, applicato alla classe Recommender, cuore del sistema, che rende possibile l'aggiunta di sue nuove estensioni in maniera semplice e immediata tramite un *override* dei metodi che garantiscono la funzionalità il filtraggio. Le modifiche al software risultano pertanto localizzate ed immediate.

In secondo luogo, il Social Web costituisce una miniera d'oro per i ricercatori che si occupano di Social Recommender System. La mole di dati da essa generata è ingente. Molti (e.g., [HNP09]) sostengono che *"More data usually beats better algorithms"*: vale a dire che, per molti problemi come quello delle raccomandazioni, non importa quanto ingegnosi siano gli algoritmi adottati: essi infatti possono essere superati avendo a disposizione, semplicemente, un quantitativo più cospicuo di dati. Ulteriori social network possono essere dunque introdotti nel sistema Cicero, allo scopo di mettere a disposizione più informazioni con cui arricchire i profili dei propri utenti. Ad esempio, per persone che possiedono account anche su Flickr o Instagram, si potrebbe pensare di integrare le annotazioni semantiche delle foto estratte da tali servizi con quelle estrapolate dalle loro bacheche su Facebook. Discorso analogo riguarda i *post*, esaminando, tra le possibili alternative, anche i *tweet* delle pagine di *Twitter* o gli aggiornamenti di stato di *GooglePlus*. Anche per questo motivo, l'architettura di Cicero è stata progettata in modo tale da ridurre al minimo le correzioni software da apportare in caso si voglia sostituire o aggiungere un nuovo social network come sorgente di dati.

Sempre relativamente alla creazione del profilo utente, la versione corrente di Cicero consente di estrarre i luoghi visitati da un individuo e dai suoi amici in un'unica occasione mediante un approccio *batch*: ciò significa che vengono letti tutti i post e le foto di Facebook pubblicati fino a quel momento, modellate le preferenze dell'utente a partire da essi e generate così raccomandazioni di beni appartenti al patrimonio artistico e culturale. Un'evoluzione interessante del progetto potrebbe prevedere la funzionalità di real time update: essa consisterebbe in un aggiornamento dell'intero grafo dei dati sociali, ogni qual volta l'account condividesse nuove risorse sul social network, con una conseguente modifica in tempo reale del profilo utente corrente e degli item raccomandati a partire da esso.

Altro possibile sviluppo futuro potrebbe riguardare i Linked Open Data. Tale mondo è vastissimo e in continuo sviluppo, a causa del numero sempre maggiore

di organizzazioni mondiali che decidono di pubblicare sul Web i propri dati in formato RDF, al fine di renderli pubblici e liberamente accessibili e interrogabili con gli standard W3C.

Ne consegue che il dominio del cultural heritage di Cicero, attualmente costituito dalle risorse fruibili su Europeana e DBpedia, può essere ulteriormente ampliato con nuovi dataset LOD, fornendo in tal modo una copertura più vasta del patrimonio culturale.

Bibliografia

[AGHT07] Jorge Aranda, Inmar Givoni, Jeremy Handcock, and Danny Tarlow. An online social network-based recommendation system. *Toronto, Ontario, Canada*, 2007.

[AGHT11] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing user modeling on twitter for personalized news recommendations. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization*, UMAP'11, pages 1–12, Berlin, Heidelberg, 2011. Springer-Verlag.

[AHHT12] Fabian Abel, Claudia Hauff, Geert-Jan Houben, and Ke Tao. Leveraging user modeling on the social web with linked data. In *Proceedings of the 12th International Conference on Web Engineering*, ICWE'12, pages 378–385, Berlin, Heidelberg, 2012. Springer-Verlag.

[BBN07] F. Bachmann, L. Bass, and R. Nord. *Modifiability Tactics*. Technical report. Carnegie Mellon University, Software Engineering Institute, 2007.

[BCK13] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice, Third Edition*. Addison-Wesley Professional, 2013.

[BLPR12] Linas Baltrunas, Bernd Ludwig, Stefan Peer, and Francesco Ricci. Context relevance assessment and exploitation in mobile recom-

mender systems. *Personal Ubiquitous Comput.*, 16(5):507–526, June 2012.

[BOHG13] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.

[CBC08] Iván Cantador, Alejandro Bellogín, and Pablo Castells. A multilayer ontology-based hybrid recommendation model. *AI Commun.*, 21(2-3):203–210, April 2008.

[CGT13] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. User-Centric vs. System-Centric Evaluation of Recommender Systems. In Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2013*, volume 8119 of *Lecture Notes in Computer Science*, pages 334–351. Springer Berlin Heidelberg, 2013.

[CP09] Li Chen and Pearl Pu. Interaction design guidelines on critiquing-based recommender systems. *User Modeling and User-Adapted Interaction*, 19(3):167–206, August 2009.

[Edm05] Diane R. Edmondson. Likert scale: A history. *Charm 2005*, 2005.

[EG04] Ralph Johnson Erich Gamma, Richard Helm. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 2004.

[EKH13] Najeeb Elahi, Randi Karlsen, and Einar J. Holsbø. Personalized photo recommendation by leveraging user modeling on social network. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, IIWAS '13, pages 68:68–68:71, New York, NY, USA, 2013. ACM.

[Eva04] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.

[Fis25] R.A. Fisher. *Statistical methods for research workers*. Edinburgh Oliver & Boyd, 1925.

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5):75 – 174, 2010.

[HB11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[HNP09] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, March 2009.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[Kru95] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Softw.*, 12(6):42–50, November 1995.

[Lar04] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[MRS09] StuartE. Middleton, DavidDe Roure, and NigelR. Shadbolt. Ontology-based recommender systems. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks

on Information Systems, pages 779–796. Springer Berlin Heidelberg, 2009.

[MSLC01] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[ONM+12] Vito Claudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, Davide Romito, and Eugenio Di Sciascio. Cinemappy: a context-aware mobile app for movie recommendations boosted by dbpedia. In Marco de Gemmis, Tommaso Di Noia, Pasquale Lops, Thomas Lukasiewicz, and Giovanni Semeraro, editors, *SeRSy*, volume 919 of *CEUR Workshop Proceedings*, pages 37–48. CEUR-WS.org, 2012.

[Pas10] Alexandre Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. AAAI, 2010.

[PCH11] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 157–164, New York, NY, USA, 2011. ACM.

[SF98] V.R. Benjamins Studer, R. and D. Fensel. Knowledge engineering: principles and methods. *IEEE Transactions on Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[SSS01] Rashmi Sinha, , Rashmi Sinha, and Kirsten Swearingen. Comparing recommendations made by online systems and friends. In *In Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 2001.

[TBLL01] James Hendler Tim Berners-Lee and Ora Lassila. The semantic web. *Scientific American*, 2001.

[WWL+13] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Tie-Yan Liu, and Wei Chen. A theoretical analysis of NDCG type ranking measures. *CoRR*, abs/1304.6480, 2013.

[ZZM+11] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Trans. Web*, 5(1):5:1–5:44, February 2011.