Data Mining Distribuito e Sistemi Multi-Agente: una panoramica

Sistemi Multi-Agente L-S

Tommaso Pirini
tommaso.pirini@studio.unibo.it

24 febbraio 2011

Sommario

Le tecnologie di Data Mining sono utili perché permettono di ricavare informazioni e trend impliciti da insiemi di dati, anche di notevoli dimensioni. Normalmente si adotta l'integrazione per generare i data warehouse che raccolgono tutti i dati in un unico luogo, su cui poi si esegue un algoritmo per estrarre un modello di predizione e una valutazione della conoscenza. Tuttavia, una singola tecnica di Data Mining non riesce ad essere appropriata per tutte le raccolte di dati e i domini, spesso caratterizzati da ampie dinamiche di cambiamento, dovute al tipo di sistema che si vuole monitorare.
Il Data Mining Distribuito nasce per far fronte al bisogno di esplorare sorgenti di dati varie e distribuite su una rete. La progettazione di questi sistemi tramite il paradigma ad Agenti è considerato un valore aggiunto.
In questo articolo viene presentata una panoramica dei sistemi di Data Mining Distribuito ad Agenti che sono ad oggi disponibili, definendo le componenti comuni, le strategie impiegate e l'architettura adottata.

Indice

1 Introduzione

Uno degli aspetti chiave dell'esplorazione di una grande quantità di sorgenti di dati, omogenee ed eterogenee, su Internet, non è soltanto il modo in cui recuperare, mettere insieme ed integrare le informazioni rilevanti, ma anche scoprire quella conoscenza preziosa che ad una prima analisi, magari superficiale, sembra nascosta, implicita, celata. Le grandi compagnie hanno capito che questa prospettiva è importante per avere entrate maggiori da applicazioni come quelle per il business data warehousing, il controllo dei processi, e la personalizzazione dei servizi per il cliente sul Web.

Nel processo di scoperta della conoscenza (*Knowledge Discovery*), che parte dall'inizializzazione dei dati per giungere alla visualizzazione delle informazioni rinvenute, il Data Mining, cioè l'estrazione (*mining*) automatica dei modelli impliciti dai dati, è l'elemento centrale. Esistono una grande varietà di tecniche di Data Mining, prodotte nelle ultime decadi del secolo scorso, di cui la maggior parte è stata sviluppata originariamente per sistemi centralizzati, cioè dove i dati sono raccolti in un unico sito. Tuttavia il bisogno crescente di poter applicare queste tecniche anche a ingenti insiemi di dati distribuiti sulla rete, ha portato alla creazione di varianti, appunto distribuite, di queste tecniche di Data Mining.

Il Data Mining Distribuito (DMD) considera differenti fonti di dati che spesso hanno realmente una diversa collocazione fisica. I dati prodotti in un specifico luogo spesso non possono essere trasferiti sulla rete per motivi legati alla velocità del trasferimento (es. *bandwith* limitata) o alla privacy, perciò è necessario che in ogni sito venga effettuata un'analisi parziale dei dati locali, ed i risultati ottenuti siano successivamente spediti ad un punto di raccolta che si occuperà di aggregarli per comporre un risultato globale.

I principali obiettivi che le tecnologie di DMD si propongono di affrontare sono [23]:

- trattare un'enorme quantità di dati (es. svariati terabyte) distribuiti su diversi nodi della rete;
- raggiungere performance temporali e spaziali accettabili, nonostante i processi computazionali intensi e la dimensione dei dati da analizzare;
- gestire la dinamicità dei dati in input, in maniera che il tempo per ricavare le informazioni necessarie non superi e sia sufficientemente inferiore alla loro effettiva validità.

Infine è importante anche considerare la sicurezza dei processi che si attuano sui dati, mantenendone l'integrità e proteggendo la privacy da intrusioni indesiderate.

Nelle sezioni successive si spiega perché viene utilizzato il paradigma ad agenti (2), lo stato dell'arte dei sistemi di DMDA (3), quali sono le componenti comuni dei sistemi di DMD (4) e quali strategie essi utilizzano (5), poi viene presentata un'architettura generale, valida per tutti questi sistemi (6) ed infine si mostrano le prospettive future di questo campo (7).

2 Perché gli Agenti?

Negli ultimi anni il paradigma ad Agenti ha trovato numerose e interessanti applicazioni in diversi domini, soprattutto nei sistemi di supporto alle decisioni e nelle applicazioni Web. In generale, le caratteristiche proprie degli agenti software, quali l'autonomia, la capacità di adattamento e di decisione, sembrano corrispondere bene ai requisiti dei sistemi distribuiti, ed in particolare di DMD.

Il concetto di scalabilità per i sistemi di DMD è fondamentale, perché la loro configurazione può cambiare in ogni istante, per cui è importante garantire la flessibilità, l'apertura e tutti i requisiti derivanti dall'ingegneria del software (*riusability*, *extensibility* e robustezza). Al fine di informare tutte le unità su ogni aggiornamento della configurazione del sistema, come ad esempio la comparsa di un nuovo host di dati, possono verificarsi richieste di interventi umani o di un meccanismo complesso che produrrebbe un forte calo delle prestazioni. Per sopperire a questi aspetti, gli agenti permettono di utilizzare tecniche come il "*collaborative learning*" [5, 25]. In questo caso infatti, gli agenti sono in grado di condividere le informazioni sulle modifiche accadute al sistema, e propagarle da un agente all'altro, consentendogli di adattarsi alla nuova configurazione aggiornata.

Un'altra proprietà importante per i sistemi di DMD è la decentralizzazione. In ogni luogo dove si raccolgono dei dati si deve poter applicare una strategia di mining specifica per un determinato dominio, o testare più di una strategia. Ogni punto di raccolta, poi, deve essere continuamente in grado di applicare ai propri dati strategie di mining provenienti dall'esterno, per ulteriori analisi, e fornire i risultati a chiunque li richieda.

Come già affermato in precedenza, ogni agente è autonomo, quindi può essere considerato come un'estensione modulare di un sistema di gestione dei dati che deliberatamente si occupa dell'accesso alla sorgente di dati sottostante, in accordo con i vincoli forniti dal sistema. Un agente però potrebbe anche non essere assegnato inizialmente ad una particolare sorgente di dati, ma potrebbe dover andare alla ricerca di quelle che rispettano alcuni criteri stabiliti, come ad esempio il tipo o la quantità dei dati da trattare. Gli agenti devono essere in grado anche di distribuirsi sui vari nodi della rete a

seconda del carico di lavoro disponibile su di essi. Ci dovranno essere quindi più agenti sui nodi che contengono una quantità di dati maggiore.

Ogni agente inoltre va programmato in una maniera il più compatta possibile, affinché possa viaggiare nella rete velocemente e senza difficoltà, al posto dei dati, solitamente più ingombranti. L'abilità di trasmettere un agente di mining da un nodo all'altro della rete permette un'organizzazione dinamica del sistema. Ad esempio, un agente $a1$ si trova sull'host $h1$ e possiede l'algoritmo di mining $m1$, sull'host $h2$ invece vi è un task che deve analizzare un insieme di dati da 1 GB e vuole usare l'algoritmo $m1$. In questa situazione conviene spostare l'agente di mining $a1$ dall'host $h1$ all'host $h2$ piuttosto che tutti i dati da $h2$ a $h1$ dove si trova l'agente in quel momento. Per approfondire ulteriormente i benefici provenienti dall'utilizzo degli Agenti per lo sviluppo dei sistemi di DMD si possono consultare i seguenti lavori [3, 22, 11, 16, 19, 20].

La sicurezza è un aspetto critico nei sistemi di DMD ad Agenti (DMDA) perché la rigidità degli schemi che ne garantiscono un buon livello inevitabilmente degrada la scalabilità del sistema [12, 26]. Le tecnologie ad agenti offrono varie soluzioni, ad esempio in [21] viene mostrata una piattaforma in cui gli agenti possono spostarsi all'interno del sistema mantenendo la privacy dei dati all'interno dei vari luoghi in cui si trovano. Le caratteristiche di auto-organizzazione permettono al sistema di non dover trasferire i dati attraverso la rete e questo li rende più protetti.

Le applicazioni del DMD sono varie: sistemi per il rilevamento delle frodi alle carte di credito, sistemi di rilevamento di intrusioni, assicurazione sanitaria, applicazioni relative alla sicurezza, clustering distribuito, segmentazione del mercato, reti di sensori, creazioni di profili della clientela, valutazione delle promozioni commerciali, analisi dei rischi di credito, ecc. In sintesi, l'utilizzo del paradigma ad Agenti può migliorare sensibilmente tutti questi sistemi di DMD perché introduce tre caratteristiche importanti:

- *interoperabilità*: non soltanto per quanto riguarda la collaborazione tra gli agenti ma anche nell'interazione con l'esterno, permettendo cioè a nuove entità di integrarsi col sistema senza difficoltà. L'architettura del sistema deve essere aperta e flessibile, e devono essere definiti protocolli di comunicazione, politiche di integrazione e un *directory service*. I protocolli di comuncazione definiscono la codifica dei messaggi, la crittografia e le techiche di trasporto degli agenti sulla rete, le politiche di integrazione invece stabiliscono come il sistema si deve comportare affinché nuovi agenti di mining o sorgenti di dati possano entrare o qualcuno di quelli già presenti possa lasciare il sistema [13, 21], il *directory service* rappresenta il servizio di "pagine gialle" del sistema.

In particolare, per quanto riguarda i protocolli di comunicazione, sono stati costituiti degli standard pubblicati dalla FIPA (*Foundation of Intelligent Physical Agents*), e la maggior parte delle piattaforme, come JADE2 e JACK3, li implementano perciò possono operare tra di loro.

- *configurazione dinamica del sistema*: è una caratteristica complessa perché è complicato pianificare e gestire gli algoritmi di mining. Per eseguire un certo algoritmo infatti potrebbero essere necessari diversi agenti e diverse sorgenti di dati. Il cambiamento dei dati influisce dunque sul processo di mining, soprattutto se i dati cambiano mentre l'agente di mining li sta ancora analizzando.

- *miglioramento delle performance*: in ambiente distribuito le attività possono essere svolte in parallelo per migliorare le performance, ricordandosi tuttavia di dover gestire la concorrenza. La qualità del servizio può essere controllata dagli agenti ma dipende anche dagli algoritmi di Data Mining stessi.

3 Lo stato dell'arte

In questa sezione forniamo una breve rassegna dei sistemi di DMDA attualmente più rappresentativi, che sono BODHI, PADMA, JAM, e *Papyrus*.

- **BODHI** [18] è stato progettato come un framework per attività di Data Mining collettive, basate su una rete contenente dati eterogenei. Su ogni postazione si trova un ambiente di esecuzione, chiamato *agent station*, che esegue il task di mining sui propri dati, e su cui gli agenti mobili possono inserire e ricavare informazioni. Il processo di mining viene distribuito quindi sia sugli agenti mobili che sulle stazioni locali, le quali posso comunicare tra di loro, per richiedere le informazioni degli agenti situati sulle stazioni vicine. Quando gli agenti si spostano infatti, portano il proprio bagaglio di conoscenza. Un *facilitator* centrale si occupa di inizializzare e coordinare tutti i processi di mining, che vengono eseguiti all'interno del sistema dalle stazioni e dagli agenti, le comunicazioni ed il traffico degli agenti.

- **PADMA** [17] si occupa del problema del DMD per le reti di dati omogenei. Il processo di mining avviene prima di tutto sviluppando i modelli dei dati locali, da parte degli agenti che risiedono su ogni nodo. La tecnica di Data Mining utilizzata in questo sistema è il *clustering* (addestramento non supervisionato). Poi, su una postazione centrale,

vengono raccolti tutti i modelli parziali, su cui si applica un ulteriore algoritmo di mining (*clustering* di secondo livello) per la generazione del modello globale.

- **JAM** [24] è un sistema multi-agente scritto in Java, progettato per utilizzare la tecnica del meta-apprendimento (*meta-learning*). Gli agenti in questo sistema possono eseguire diverse tecniche di classificazione (Ripper, CART, ID3, C4.5, Bayes e WEPBLS) su database (relazionali) eterogenei. Inizialmente, ogni agente, sia che risieda sul nodo locale sia che sia stato importato dai nodi vicini, costruisce un classificatore secondo la tecnica di mining che implementa. Ogni agente implementa una sola tecnica di classificazione. Il sistema, successivamente, attraverso un insieme di agenti di meta-apprendimento combina questi modelli, sviluppati nei singoli nodi, in un unico classificatore. Una volta che il classificatore globale è stato completato, il sistema coordina l'esecuzione di una serie di test, basati su differenti insiemi di dati, in maniera simultanea ed indipendente, per calcolarne l'accuratezza. In molti casi l'accuratezza globale delle predizioni risulta migliore rispetto ai classificatori locali.

- **Papyrus** [4] è un sistema anch'esso scritto in Java, progettato per il *clustering* su reti di dati eterogenei e per il *meta-clustering*. Gli agenti mobili del sistema possono spostare, da un nodo all'altro della rete, sia i modelli di conoscenza, costruiti localmente per ridurre il traffico di rete, sia i dati veri e propri, per ottenere un risultato globale. La coordinazione di questo sistema può essere sia centralizzata su un unico nodo che distribuita (peer-to-peer) su differenti nodi. *Papyrus* supporta vari metodi di scambio delle informazioni, ma che devono essere descritti utilizzando uno speciale linguaggio di markup fornito dal sistema stesso.

Quello che accomuna tutti gli approcci appena descritti è che la conoscenza ricavata dai dati, situati sui differenti nodi della rete, viene sempre raccolta e integrata, cercando di minimizzare il traffico di rete e di massimizzare la computazione locale. Inoltre, i sistemi di DMDA coprono tutte le possibili reti di dati, sia omogenee che eterogenee, e i database, sia multipli che distribuiti.

4 Le componenti principali del sistema

La struttura del sistema di DMDA può essere descritta da un insieme di componenti visibile in Figura 1. Analizziamo ora le due attività principali

del sistema, richiesta e risposta, ciascuna delle quali coinvolge un insieme differente di componenti.

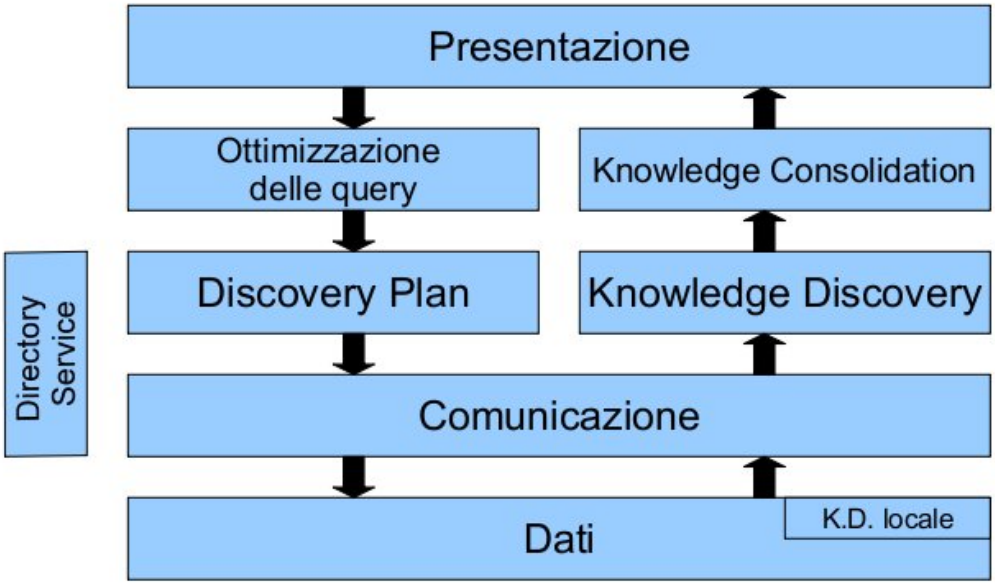

Figura 1: Componenti di un sistema DMDA.

Alla base della struttura ci sono i **dati** che, a seconda del loro scopo, possono essere contenuti in diversi nodi e in varie forme (database relazionali, stream, pagine web, ecc.). Sopra il livello dei dati c'è il livello di **comunicazione** in cui il sistema sceglie le risorse dal *directory service*, il quale mantiene una lista di tutto ciò che è presente nel sistema, come le sorgenti di dati, gli algoritmi di mining, i tipi di dati, ecc. I protocolli di comunicazione dipendono dalla particolare implementazione del sistema (client-server, peer-to-peer,...).

Nel punto più alto invece c'è il livello di **presentazione** che si occupa di fornire un'interfaccia per l'interazione con gli utenti, tramite tool di reporting visuali, messaggi comprensibili dagli utenti, form da compilare, ecc.

Quando un utente, attraverso l'interfaccia, effettua una richiesta di mining al sistema sono coinvolte le seguenti componenti: **Ottimizzazione delle query** e **Discovery Plan**. Il primo analizza la richiesta che è stata effettuata per indicare quale algoritmo di mining si deve utilizzare e su quali sorgenti di dati farlo. Esso determina anche se è possibile eseguire in concorrenza i task, nel caso i dati fossero distribuiti su diversi siti e possano essere analizzati parallelamente. Il secondo invece alloca i sotto-task alle relative risorse. In questo livello gli agenti mediatori svolgono un ruolo molto im-

portante: coordinano le unità di computazione, le quali eseguono la propria parte di mining su ogni nodo in maniera asincrona, e raccolgono i risultati dei task.

Quando invece si devono eseguire gli algoritmi e successivamente mostrare i risultati ottenuti, i componenti che intervengono sono: **Knowledge Discovery** e **Knowledge Consolidation**. In particolare, per ogni nodo, può intervenire anche una **Knowledge Discovery locale** che ricava un modello locale dai propri dati e lo trasmette sulla rete. Questo accade nel caso in cui il processo di mining avviene localmente perché dipendente dalla particolare struttura dei dati. In generale però, l'algoritmo di mining vero e proprio viene eseguito dalla Knowledge Discovery, la quale analizza i dati provenienti da una sorgente per ottenere un modello di conoscenza. La Knowledge Consolidation infine raccoglie tutti i modelli ottenuti dai vari nodi e li compatta in un'unico risultato significativo, da presentare all'utente. Spesso questo livello è molto complesso perché in alcuni casi risulta difficile armonizzare i risultati provenienti da sorgenti diverse, a volte molto eterogenee.

5 Le strategie adottate

I sistemi di DMD che sono stati sviluppati fino ad oggi si possono raggruppare in tre categorie, in base alla strategia da loro utilizzata [23].

5.1 Strategia di *Central Learning*

Questa strategia si applica quando i dati possono essere raccolti in un solo luogo e può essere costruito un unico modello. È necessario un solo requisito: essere in grado di spostare tutti i dati in una unica postazione per applicarvi uno o più algoritmi di Data Mining sequenziali. Questa strategia viene utilizzata soprattutto quando l'area di distribuzione dei dati è ristretta e i dati da analizzare non sono numerosi. È una soluzione molto costosa ma anche la più accurata.

L'accentramento dei dati non è una semplice operazione di raccolta poiché coinvolge anche logiche di associazione, come nel caso in cui alcuni dati siano su un host ed alcuni attributi relativi a quegli stessi dati siano su un altro host differente. Non può comunque considerarsi una strategia sempre realizzabile [7] ed l'impiego di una tecnologia ad agenti in questo caso non porterebbe grossi miglioramenti.

5.2 Strategia di *Meta-Learning*

È una strategia che costruisce dei classificatori da quantità di dati distribuite in maniera omogenea. Si compone di tre passi fondamentali. Il primo consiste nel generare dei modelli locali in ogni postazione su cui ci sono dei dati, usando un algoritmo di classificazione. Nel secondo i classificatori locali vengono raccolti in un unica postazione, e vengono valutati su un nuovo insieme di dati. Nel terzo ed ultimo passo si costruisce il classificatore finale, tramite una composizione di quelli precedenti o selezionando il migliore.

In questo caso su ogni nodo della rete dovrà essere presente una copia di agente classificatore. Due esempi di questo tipo di sistemi sono JAM e BODHI [8].

5.3 Strategia di *Hybrid Learning*

Questa strategia realizza un modello di conoscenza combinando l'apprendimento locale a quello centralizzato [9]. *Papyrus* [10] è, ad esempio, un sistema che supporta entrambe le strategie. Contrariamente a JAM e BODHI, è in grado di trasferire da un nodo all'altro sia modelli sia dati, a seconda della strategia adottata.

Il problema maggiore di questo genere di sistemi è che non sempre si ottiene il risultato finale esatto, cioè il modello di conoscenza globale, ricavato da un algoritmo distribuito, potrebbe essere differente da quello che si otterrebbe dall'esecuzione dello stesso algoritmo centralizzato, applicato ai medesimi dati. Una certa approssimazione dei risultati potrebbe non essere un problema in alcuni domini, ma è comunque importante tenere conto di questo fatto.

Altri problemi sono dati dai cosiddetti "colli di bottiglia" dell'hardware: infatti non sempre tutte le risorse fisiche utilizzate hanno le stesse prestazioni, e se una parte della computazione avviene localmente, le performance generali dipenderanno fortemente dal nodo più lento, quello cioè che ci metterà più tempo a computare il suo algoritmo di mining. Infine è importante considerare anche il fatto che i modelli ricavati da database con schemi differenti siano incompatibili.

Nel DMD c'è una stretta correlazione tra accuratezza e costi di computazione. Da una parte, se il nostro scopo fosse mantenere bassi i costi di computazione e di trasferimento, dovremmo processare localmente i dati e poi combinare i risultati locali in un unico risultato globale. In questa maniera, il traffico sarà il minore possibile, c'è però il rischio di perdere accuratezza. Dall'altra parte invece, se vogliamo ottenere il risultato più preciso possibile,

dovremmo raccogliere tutti i dati in un solo nodo analizzandoli poi con un algoritmo di mining sequenziale. In questo caso avremmo dunque la massima accuratezza, ma anche il massimo costo in termini di traffico di rete.

6 Un'architettura generale

La maggior parte dei *framework* per DMDA adotta una architettura simile a quella mostrata in Figura 2 e fornisce componenti strutturali comuni [17, 18]. Inoltre per facilitare le interazioni tra gli agenti si utilizza un linguaggio di comunicazione standard, come KQML o FIPA-ALC.
Qui di seguito si elencano i tipi più comuni di agenti presenti in queste architetture, anche se alcuni nomi potrebbero risultare differenti, nella maggior parte dei casi, condividono le stesse funzionalità.

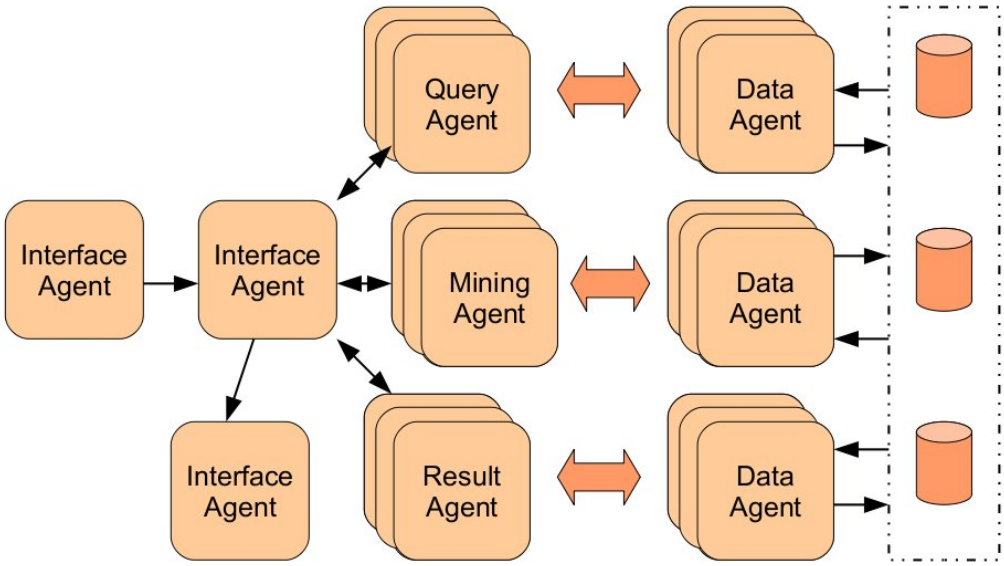

Figura 2: Architettura generale di un sistema DMDA.

Interface Agent (o User Agent): è l'agente che interagisce con l'utente umano o un altro agente "user", raccogliendo le sue richieste e fornendo in risposta i risultati finali, magari attraverso una visuale grafica (tabelle, grafi,...). La sua interfaccia contiene i metodi per ricevere gli input dall'utente e per comunicare con gli altri agenti, in particolare col *facilitator*. Egli si occupa anche di salvare nel suo modello di co-

noscenza l'elenco delle interazioni che ha con i vari utenti, creando dei profili con le loro preferenze.

Facilitator Agent (o Manager Agent): è il responsabile dell'attivazione e della sincronizzazione degli altri agenti. Inizialmente riceve una richiesta dall'*Interface Agent*, quindi elabora un piano di lavoro che cerca di portare a compimento. Per fare ciò può chiedere l'intervento di altri agenti, poi, una volta ottenuti tutti i risultati, li sintetizza e trasmette l'esito all'*Interface Agent* per la visualizzazione. È quindi anch'esso in grado di comunicare con gli altri agenti e di gestire e coordinare vari task. Il piano di lavoro che egli stabilisce, dopo aver ricevuto una richiesta, viene costruito in base a delle specifiche "ontologie" contenute nel suo modello di conoscenza. Per il fatto che è in grado di richiedere un servizio ad altri agenti, esso possiede anche una meta-conoscenza che gli permette di riconoscere le loro capacità.

Data Agent (o Resource Agent): è l'agente che contiene le informazioni relative a ciascuna delle sorgenti di dati (meta-dati) ed è quindi responsabile della preparazione e della consegna di tali dati agli agenti di mining, per la loro analisi. Si occupa inoltre di risolvere i conflitti nella definizione e rappresentazione dei dati e di gestire l'eterogeneità dei database. La sua interfaccia contiene i metodi per comunicare con gli altri agenti e con le sorgenti di dati. In particolare, su queste ultime, effettua le query per il recupero dei dati che fornisce poi al *facilitator* o ad altri agenti di mining.

Mining Agent: è l'agente che implementa una specifica tecnica o algoritmo di Data Mining. Una volta portato a termine il suo lavoro trasmette i risultati al *facilitator* o ai *Result Agent*: supporta quindi anche la comunicazione con gli altri agenti. Il suo modulo della conoscenza contiene meta-conoscenza che riguarda le tecniche di mining, come essere in grado di scegliere quale metodo è adatto per quale tipo di problema, quali input servono a ogni algoritmo di mining, quali formati di dati può riceve in input, ecc.

Result Agent: si occupa di recuperare i risultati elaborati dagli agenti di mining. Una volta raccolti tutti i risultati li unifica e li dispone, insieme al *facilitator*, per mostrarli all'utente. La sua interfaccia contiene i metodi per ricevere i risultati dei processi di mining dagli agenti, comunicare col *facilitator* e gestire le rappresentazioni grafiche con cui presentare i risultati finali. Il suo modello di conoscenza è quindi in grado di salvare i particolari relativi ai template di reportistica e le

primitive di visualizzazione che vengono usate per presentare i risultati all'utente.

Broker Agent (o Matchmaker Agent): serve da consulente per favorire la diffusione delle richieste agli agenti che sono in grado di occuparsene. Esso è come un servizio di "pagine gialle" che risponde ai bisogni dei *facilitator* restituendo il nome e l'ontologia di un agente che ha le capacità richieste. Quando un nuovo agente di mining vuole dunque entrare a far parte del sistema deve registrarsi presso il *broker*, che è colui che tiene in memoria i nomi, le ontologie e le capacità di ogni altro agente facente parte del sistema.

Query Agent: generato ad ogni richiesta dell'utente fatta al sistema. Il suo modulo della conoscenza contiene i meta-dati relativi alle strutture dei dati, locali e globali. Queste informazioni vengono utilizzate poi nella generazione delle query per recuperare i dati richiesti.

Ontology Agent: contiene un elenco globale di tutte le ontologie presenti nel sistema e risponde alle domande su di esse. Il suo comportamento potrebbe semplicemente essere quello di tenere in memoria le ontologie registrate, ma potrebbe anche essere in grado di utilizzare un ragionamento semantico, per determinare l'applicabilità di un dominio ad una particolare richiesta di mining.

Mobile Agent: è un agente che viaggia attraverso la rete. In ogni luogo che visita, processa i dati locali e spedisce i risultati all'host che li ha richiesti, invece di far viaggiare sulla rete i dati "grezzi". Ha il vantaggio di generare minor traffico, perché trasmette sulla rete i risultati al posto dei dati. In generale, evitare di trasmettere i dati consente anche una maggior sicurezza per quanto riguarda la privacy, soprattutto se i dati da analizzare sono sensibili. Si deve comunque avere la garanzia che il comportamento dell'agente che viaggia tra i nodi sia corretto, e non nasconda insidie per gli host e la loro sicurezza. Per utilizzare questi agenti è inoltre necessario installare una piattaforma per la loro gestione su ogni nodo della rete.

Local Task Agent: è l'agente che risiede su ciascun nodo della rete. Il suo compito è registrare al *facilitator* le informazioni relative all'host che rappresenta, e rispondere alle richieste degli agenti di mining. Esso può inoltre recuperare i dati salvati sul proprio nodo, elaborarli e riportare i risultati nel sistema.

KDD System Agents: sono agenti che, in alcuni sistemi di DMDA, gestiscono l'intero processo di Knowledge Discovery sui dati. Tra di loro ci sono le seguenti categorie:

- **Pre-Processing Agent:** prepara i dati per il mining. A seconda dello specifico algoritmo di mining ed alle sue necessità, effettua una selezione ed una pulitura dei dati.
- **Post Data Mining Agent:** valuta le performance, l'accuratezza ed altre statistiche relative all'elaborazione dei risultati, effettuata dagli agenti di mining.

7 Le prospettive future

Questa integrazione tra due tecnologie, il paradigma ad Agenti e il Data Mining, potrebbe portare un'evoluzione nello sviluppo dei processi produttivi e nell'utilizzo dei sistemi software, in varie direzioni. Proviamo ad analizzarne alcune.

7.1 Prospettive della ricerca

Le applicazioni che vogliono fare DMD devono innanzitutto preoccuparsi di analizzare gli aspetti che riguardano la computazione distribuita, la comunicazione tra le parti e la memorizzazione della conoscenza del sistema.

Per sviluppare sistemi di DMDA ci si è anche ispirati ai fenomeni naturali. Di recente, alcuni ricercatori hanno implementato sistemi di DMD basandosi sulla *Swarm Intelligence* [2, 6], una proprietà strettamente correlata con il paradigma ad Agenti. Tra questo tipo di applicazioni ci sono classificatori basati su regole logiche, selezionatori di percorsi tramite *Ant Colony Optimization*, Text Mining con entità di clustering gerarchiche, ecc.

7.2 Prospettive dell'ingegneria del software

Una caratteristica importate dei sistemi di DMDA è lo scambio di modelli di conoscenza tra gli host della rete. Per fare ciò in maniera trasparente e senza difficoltà è necessario rendere standard la rappresentazione e lo scambio di tali modelli (es. PMML [1]). È utile anche disporre di tool di supporto alla progettazione di applicazioni di Data Mining e database distribuiti, come ad esempio CRISP-DM [1].

Si deve considerare inoltre, il fatto che attualmente, le sorgenti di dati interessanti per questo tipo di analisi non si limitano più soltanto ai classici

database relazionali. Esistono infatti un'ampia gamma di sorgenti di dati, spesso di dimensione molto piccola, come quelle costituite dalle memorie degli smartphone, che risulta molto conveniente poter integrare nei sistemi di DMDA. In questo caso si possono utilizzare degli agenti mobili, i quali possono essere scaricati sul dispositivo, eseguire il proprio task, e restituire ad un sistema centrale una rappresentazione del modello di conoscenza rilevato, per ulteriori analisi.

Un'altro ingrediente è l'emergere delle Architetture Orientate ai Servizi (SOA), che favoriscono l'integrazione di applicazioni ad agenti in maniera molto vantaggiosa. Inoltre le applicazioni Web stanno diventando ogni giorno maggiormente popolari ed Internet sta diventando una componente sempre più necessaria in ogni sistema moderno.

7.3 Prospettive dei sistemi

Una nuova prospettiva per l'integrazione tra Data Mining e Agenti è data dai sistemi mobili, pervasivi e peer-to-peer (P2P). Una caratteristica particolare di questi sistemi di computazione è che operano con un insieme di sorgenti di informazioni notevolmente dinamico. Ad esempio, i dispositivi mobili possono spostarsi, entrare ed uscire, liberamente da una rete all'altra: come cambia la rete e quindi la topologia di comunicazione, così cambiano anche i servizi a loro disposizione. In ambienti come *smart space* o *ambient intelligence*, le decisioni vengono prese in base all'unione delle informazioni ricevute dai sensori distribuiti, e dai dispositivi mobili, che pervadono l'ambiente. Uno degli obiettivi, in questo senso, è quello di riuscire a rendere un ambiente adatto alle abitudini e alle preferenze degli esseri umani che li vivono, tramite la creazione e l'aggiornamento dei loro profili. Il paradigma ad Agenti si adatta bene alle esigenze di questo genere di applicazioni, sia per quanto riguarda l'architettura sia per le tecnologie di progettazione [14].

Una caratteristica della maggior parte degli attuali sistemi P2P è quella di essere notevolmente scalabili. Gli algoritmi P2P non dipendono da un server centrale, ma ogni host esegue il proprio task e richiede agli altri i dati di cui ha bisogno, se questi sono disponibili. Tuttavia la sicurezza è una caratteristica critica anche per questo tipo di sistemi, perché lo scambio di informazioni con gli altri nodi della rete può creare vulnerabilità al sistema. Gli host potrebbero infatti avere comportamenti "egoistici", come richiedere agli altri le informazioni senza trasmettere le proprie. Per questo prima di entrare nella rete ogni host deve accettare i termini e le condizioni d'uso stabiliti per tutti da un protocollo [8, 15].

7.4 Prospettive dell'utente

Infine per quanto riguarda le caratteristiche dell'interazione con l'utente dei sistemi di DMD si desiderano supporti a livello di sistema per l'interazione di gruppo e per il collaborative problem solving, lo sviluppo di interfacce-utente alternative (soprattutto per i dispositivi mobili) e la gestione dei requisiti di sicurezza.

Riferimenti bibliografici

[1] The Data Mining Group (DMG). `http://www.dmg.org/`.

[2] A. Abraham, C. Grosan, and V. Ramos. *Swarm intelligence in data mining*. Springer-Verlag New York Inc, 2006.

[3] S. Baik, J. Bala, and J. Cho. Agent based distributed data mining. *Parallel and Distributed Computing: Applications and Technologies*, pages 185–199, 2005.

[4] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: a system for data mining over local and wide area clusters and super-clusters. In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, page 63. ACM, 1999.

[5] A. Bordetsky. Agent-based support for collaborative data mining in systems management. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, page 9. IEEE, 2002.

[6] S.A. Brueckner and H. Van Dyke Parunak. Swarming distributed pattern detection and classification. *Environments for Multi-Agent Systems*, pages 232–245, 2005.

[7] J.C. Da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch. Distributed data mining and agents. *Engineering Applications of Artificial Intelligence*, 18(7):791–807, 2005.

[8] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in peer-to-peer networks. *Internet Computing, IEEE*, 10(4):18–26, 2006.

[9] W. Davies and P. Edwards. Distributed learning: An agent-based approach to data-mining. In *Proceedings of Machine Learning 95 Workshop on Agents that Learn from Other Agents*. Citeseer, 1995.

[10] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[11] C. Giannella, R. Bhargava, and H. Kargupta. Multi-agent systems and distributed data mining. *Cooperative Information Agents VIII*, pages 1–15, 2004.

[12] V. Gorodetski and I. Kotenko. The multi-agent systems for computer network security assurance: frameworks and case studies. 2002.

[13] V. Gorodetskiy, O. Karsaev, V. Samoilov, and S. Serebryakov. Agent-based Service-Oriented Intelligent P2P Networks for Distributed Classification. In *Hybrid Information Technology, 2006. ICHIT'06. International Conference on*, volume 2, pages 224–233. IEEE, 2006.

[14] V. Gorodetsky, O. Karsaev, V. Samoylov, and S. Serebryakov. Interaction of agents and data mining in ubiquitous environment. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 3, pages 562–566. IEEE, 2009.

[15] V. Gorodetsky, O. Karsaev, V. Samoylov, and S. Serebryakov. P2P Agent Platform: Implementation and Testing. *Agents and Peer-to-Peer Computing*, pages 41–54, 2010.

[16] V. Gorodetsky, O. Karsaeyv, and V. Samoilov. Multi-agent technology for distributed data mining and classification. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 438–441. IEEE, 2003.

[17] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining using an agent based architecture. Technical report, Los Alamos National Lab., NM (United States), 1997.

[18] H. Kargupta, B.H. Park, D. Hershberger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. *Advances in Distributed and Parallel Knowledge Discovery*, pages 133–184, 1999.

[19] M. Klusch, S. Lodi, and M. Gianluca. The role of agents in distributed data mining: Issues and benefits. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 211–217. IEEE, 2003.

[20] M. Klusch, S. Lodi, and G. Moro. Issues of agent-based distributed data mining. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, page 1035. ACM, 2003.

[21] X. Li and J.B. Ni. Deploying mobile agents in distributed data mining. *Emerging Technologies in Knowledge Discovery and Data Mining*, pages 322–331, 2009.

[22] H.V.D. Parunak and S.A. Brueckner. Engineering swarming systems. *Methodologies and Software Engineering for Agent Systems*, 24, 2004.

[23] V.S. RAO. MULTI AGENT-BASED DISTRIBUTED DATA MINING: AN OVER VIEW. 2009.

[24] S. Stolfo, A.L. Prodromidis, S. Tselepis, W. Lee, D.W. Fan, and P.K. Chan. JAM: Java agents for meta-learning over distributed databases. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 74–81, 1997.

[25] J. Tožička, M. Jakob, and M. Pěchouček. Market-inspired approach to collaborative learning. *Cooperative Information Agents X*, pages 213–227, 2006.

[26] F.E. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.