

GOVERN MEANING

L'ontologia è
l'infrastruttura invisibile
che definisce la realtà
per AI, agenti e persone.

Chi decide cosa significa
ogni cosa, decide
come verrà ragionato,
automatizzato e governato
il futuro.

// ai strategy papers

idee, strategie e impatti
dell'intelligenza artificiale



Ontologia e knowledge graph: perché la tua azienda ne ha bisogno (e come si fa)

Tutti parlano di ontologia, quasi nessuno la governa. Eppure è la struttura che decide se la tua AI ragiona o indovina, se una risposta è difendibile davanti a un regolatore o è solo plausibile. Guida lunga e ragionata: cos'è davvero un'ontologia, perché serve adesso, come si costruisce senza reinventarla, e come si evita che degradi in silenzio fino a diventare pericolosa.

Prendi la parola "cliente". In una riunione dove siedono vendite, finanza e supporto, quella parola significa tre cose diverse allo stesso tavolo. Per il commerciale è chiunque sia entrato nella pipeline, anche solo un contatto che ha lasciato un'email. Per la finanza è un'entità fatturata, con partita IVA e uno storico di pagamenti. Per il supporto è qualcuno che ha diritto ad aprire un ticket. Tre definizioni, tre reparti, un'unica parola, e ogni report costruito su quella parola porta dentro l'ambiguità senza dichiararla.

Finché a leggere quei report sono persone, il malinteso si assorbe. Un analista sa di quale "cliente" sta parlando il collega, aggiusta a mente, chiede conferma. Il problema nasce quando a leggere i dati mettiamo una macchina che deve rispondere, decidere, agire. La macchina non aggiusta a mente. Prende la definizione che trova, la applica con sicurezza, e ti restituisce un numero di retention, un'esposizione al rischio, una lista di destinatari di una campagna, senza sapere che quella definizione ne conteneva altre due dentro.

È da qui che parte tutta la storia, e negli ultimi mesi è finita su ogni slide, in ogni pitch, dentro quasi ogni conversazione su come portare l'intelligenza artificiale in azienda. La parola che gira è "ontologia", accompagnata dalle sue compagne, knowledge graph, semantic layer, grafo della conoscenza. Vengono pronunciate come se il significato fosse ovvio e condiviso, e quasi mai lo è. La si potrebbe [riassumere così](#): adesso tutti hanno un'ontologia, e quasi nessuno ha un'ontologia.

Vorrei prendere quella frase sul serio, e spiegare a chi in azienda deve decidere, non solo implementare, cosa c'è sotto queste parole, perché il tema esplode proprio ora, e cosa cambia nel modo di lavorare quando l'ontologia smette di essere una parola d'ordine e torna a essere quello che è, la struttura del significato su cui poggia tutto il resto.

01 La parola svuotata

Faccio spesso un gioco, dai tavoli dove lavoro, quando qualcuno tira fuori la parola ontologia con l'aria di chi ha trovato la soluzione. Chiedo: quale problema stiamo resolvendo, esattamente. Nove volte su dieci la risposta è una seconda parola d'ordine al posto di una definizione, ci serve per l'AI, ci serve per mettere ordine nei dati, ci serve perché il competitor la sta facendo.

La stessa scena, [vista da dentro un workshop](#), la racconta chi di architetture software vive: qualcuno nomina l'AI, qualcun altro parla di API, e sul tavolo finiscono mischiati tre concetti che non hanno niente a

che vedere, Canonical Data Model, ontologia e knowledge graph, che alla domanda se siano la stessa cosa non lo sono neanche lontanamente, e la confusione ha un costo preciso, over-engineering, sistemi che provano a fare tutto insieme e finiscono per non fare bene niente.

Il Canonical Data Model è un modello dati canonico, serve a far parlare tra loro sistemi diversi standardizzando i formati di scambio. Vive nel mondo dell'integrazione, non della conoscenza. Un'ontologia è un'altra cosa, la formalizzazione dei concetti di un dominio, delle loro proprietà, delle relazioni ammesse e delle regole che li vincolano. Il knowledge graph è un'altra cosa ancora, quella struttura riempita di dati reali, la rete concreta di istanze. Confonderli ha un costo concreto, progetti che partono storti e si scoprono storti dopo mesi, quando il conto è già salato.

L'anti-pattern ha perfino un nome, il modello unico per dominarli tutti, e descrive la tentazione di costruire un solo modello universale che assorbe integrazione, semantica e conoscenza, e finisce per non fare bene nessuna delle tre. Contro quella tentazione tengo in tasca una domanda che è la forma più economica di due diligence che conosca, e la ripeto in ogni riunione dove si propone di risolvere tutto con un knowledge graph: quale problema stiamo davvero cercando di risolvere. Se la risposta è un elenco di tre cose diverse a livelli diversi, probabilmente servono tre strumenti, non un modello solo che prova a essere tutti e tre.

Poi arriva il marketing, e la nebbia si infittisce. Sotto l'etichetta ontologia oggi si vende un po' di tutto: Microsoft con Fabric IQ, Palantir con la sua Ontology, e a seguire SAP, Salesforce, AWS, ciascuno con la propria versione della stessa narrativa. Quasi sempre, dietro l'etichetta, si trova un grafo di proprietà governato, o un glossario di business con qualche tipo di relazione, o uno schema tipizzato con dei vincoli. Cose utili, a volte molto sofisticate. Ma non ontologie nel senso logico del termine, e la distinzione pesa enormemente nel momento in cui un agente autonomo comincia a ragionarci sopra per prendere decisioni operative.

La differenza sta in cosa manca, ed è lì che si nasconde il rischio. Un'ontologia formale, nel senso pieno, si impegna su una semantica logica fatta di assiomi, prende posizione esplicita su come trattare ciò che non è stato dichiarato, distingue individui e classi, e soprattutto fa inferenza: da ciò che hai asserito un motore deriva ciò che non hai scritto, in modo limitato, decidibile, verificabile. [Guardate dentro le offerte dei vendor](#) con questa lente e il quadro cambia. Fabric IQ nasce da un modello semantico di Power BI disegnato da esperti di business con strumenti visuali senza codice. La Ontology di Palantir definisce tipi di oggetto, di collegamento e di azione, ed è la più sofisticata del gruppo, ma sotto non ha logica descrittiva, non ha inferenza, non prende un impegno formale su un regime epistemologico. Il risultato è una mappa concettuale ben governata, la promessa è ragionamento semantico, e le due cose non coincidono. La posta non è accademica: quello stesso strato viene posizionato come fondamento per agenti che decidono in autonomia, sostiene operazioni di difesa e infrastrutture critiche, connette i dati di una rete elettrica. Un agente che agisce da solo su un livello semantico informale ragiona bene solo quanto è completo un modello costruito a mano, senza inferenza né validazione formale dei vincoli. Non puoi certificare il comportamento di un agente contro vincoli che non hai mai specificato, e un'infrastruttura costruita su una scommessa di convergenza semantica resta fragile e opaca, e fallisce in silenzio finché non fallisce di colpo.

La domanda da rivolgere a chi ti vende un'ontologia, allora, è secca, e la tengo pronta per la prossima riunione: cosa può *dimostrare*, esattamente, il vostro sistema. Non cosa può rappresentare, non cosa mostra in una demo. Quali garanzie formali offre quando un agente deriva una conclusione che nessuno aveva scritto. Se la risposta è vaga, ti stanno vendendo una mappa ben disegnata, non un motore di

ragionamento. Magari va benissimo lo stesso, però è bene sapere quale delle due cose stai comprando, perché il prezzo e i rischi sono diversi.

02 Ontologia, tassonomia, knowledge graph

Facciamo ordine dal basso, salendo per gradi, perché ogni gradino di questa scala è un concetto che nei pitch si salta e che invece decide tutto.

Il mattone elementare si chiama [tripla](#), e viene dallo standard [RDF del W3C](#), nato per rappresentare informazione sul web in modo leggibile dalle macchine. Ha tre parti, soggetto, predicato e oggetto, come in "Mario, vive in, Italia" o in "Prodotto A, ha caratteristica, impermeabile", e con questi tre elementi ottieni un'affermazione atomica, precisa, interrogabile. Metti insieme milioni di triple e hai qualcosa che somiglia a conoscenza, a patto che usino gli stessi identificatori per le stesse cose, e qui entra un dettaglio che sembra tecnico ed è invece politico: gli IRI, gli identificatori univoci che evitano di confondere nello stesso nodo il "Paris" città e il "Paris" persona. È lo stesso lavoro dell'annotazione semantica, marcare in un testo la parola "Paris" come città e non come persona, e separa i dati che una macchina attraversa a casaccio dai dati che sa leggere. Attorno a RDF ruotano altri standard da nominare una volta per non confonderli, SKOS per organizzare vocabolari e tassonomie, OWL per le ontologie vere con le loro regole, e il collegamento tra dataset diversi che va sotto il nome di linked data. Sono i mattoni con cui il web semantico prometteva, vent'anni fa, di rendere i dati comprensibili alle macchine, e tornano utili adesso per una ragione che allora non c'era, dare a un modello linguistico qualcosa di solido su cui poggiare invece del vuoto.

Sopra le triple sta la tassonomia, la gerarchia, l'albero delle categorie, la relazione "è un tipo di". Un cane è un tipo di mammifero, un mammifero è un tipo di animale. La [distinzione lessicale più chiara](#) che ho trovato la prendo in prestito e la tengo. La tassonomia è la gerarchia. L'ontologia è la gerarchia più le relazioni più le regole più i vincoli su cosa può connettersi a cosa. Una frattura riguarda un osso, e questa è una regola che vive nell'ontologia, non nella tassonomia. Il knowledge graph è l'ontologia popolata con i dati veri, i milioni di istanze reali che scorrono nei tuoi sistemi.

La scala dell'astrazione

Dal dato grezzo al governo del significato. Ogni gradino aggiunge una capacità che quello sotto non ha.

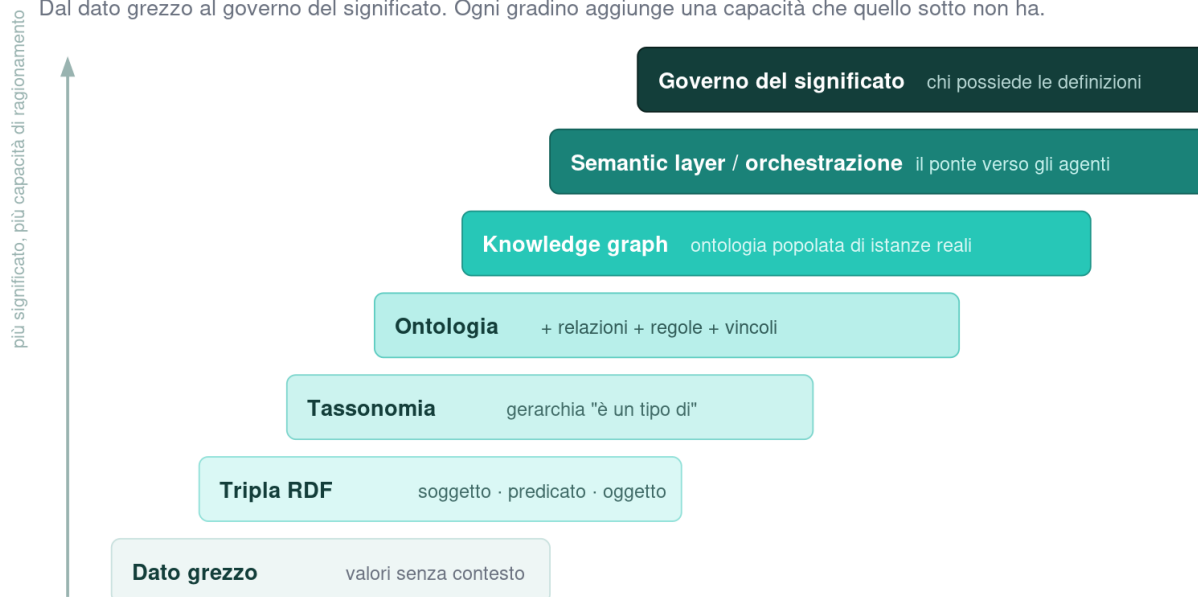


Fig. 1 – La scala dell'astrazione, dal dato grezzo al governo del significato.

La definizione classica di ontologia, quella che citano tutti e che pochi hanno letto per intero, la diede Tom Gruber negli anni Novanta: [una specificazione esplicita di una concettualizzazione](#). Suona astratto e dice una cosa semplice e potente. Un'ontologia è il manuale d'istruzioni con cui una macchina capisce di cosa stiamo parlando. Il manuale si scompone in cinque componenti da tenere a mente: le classi (Persona, Azienda, Città), le istanze concrete (Mario Rossi), le proprietà (età 35), le relazioni (lavora per), i vincoli (la data di fondazione deve precedere quella di scioglimento). Bastano questi cinque pezzi perché una macchina smetta di vedere testo e cominci a vedere struttura.

Ontologia e knowledge graph

L'ontologia è lo schema, il grafo è quello schema riempito di dati reali.

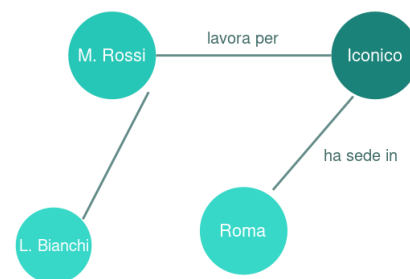
ONTOLOGIA · lo schema

tipi, relazioni ammesse, regole



KNOWLEDGE GRAPH · le istanze

gli stessi tipi, riempiti di dati veri



milioni di nodi e archi tipizzati,
tutti conformi allo schema di sinistra.

Fig. 2 – L'ontologia e lo schema, il knowledge graph e le sue istanze.

Ne manca uno, ed è quello che separa chi ha studiato la materia da chi la nomina soltanto. Un'ontologia porta dentro di sé degli impegni, i commitment ontologici, decisioni su cosa esiste, su come l'identità di una cosa persiste nel tempo, su cosa conta come relazione primitiva. L'esempio più citato in questo campo lo trovo illuminante per chiunque debba modellare dati aziendali. Una persona è un tipo, un kind: fornisce identità, è rigida, esiste in quanto tale. Uno studente è un ruolo, un role: dipende dal contesto, è definito da fuori, e una persona può smettere di essere studente senza smettere di esistere. Modellare i due come se fossero la stessa categoria produce, mesi dopo, confusione nel ciclo di vita dei dati, fallimenti di interoperabilità, quella deriva silenziosa del significato che si scopre solo quando è tardi. Un modello logicamente valido può essere ontologicamente sbagliato, ed è una frase che chi porta l'AI in azienda dovrebbe stamparsi da qualche parte.

Sopra tutto stanno le upper ontology, le ontologie fondazionali che rispondono a una domanda sola, quali impegni strutturali servono perché una rappresentazione conti come conoscenza coerente del mondo. Ne esistono di diverse e non concordano tra loro. BFO, fortemente realista, descrive la realtà così com'è e fa da spina dorsale a mezza biomedicina. DOLCE guarda a come gli umani concettualizzano il mondo. UFO, col suo linguaggio OntoUML, tiene insieme ontologia formale e ingegneria del software. Più che vocabolari in competizione, sono impegni metafisici in competizione. Al decisore non serve scegliere tra BFO e DOLCE, gli serve sapere che la scelta esiste, che qualcuno la sta facendo per lui, e che farla male non dà un errore a schermo: dà un sistema che funziona finché non funziona più.

Un esempio solo, per far sentire il peso di questi impegni, su una domanda che l'industria si pone ogni giorno. La versione A di un prodotto, la versione B, il sistema in manutenzione e quello dismesso sono la stessa cosa, oppure no. Una tradizione filosofica, l'endurantismo o vista tridimensionale, dice che l'oggetto è interamente presente in ogni istante e il cambiamento si rappresenta come una successione di stati, col tempo che resta fuori. Un'altra, il perdurantismo o vista quadridimensionale, dice che l'entità si estende nel tempo come nello spazio, e "l'auto di ieri" e "l'auto di oggi" sono fette temporali della stessa entità estesa nel tempo. Sembra un dibattito da seminario, e invece decide come gestisci identità,

tracciabilità, gemelli digitali, versioning, responsabilità legale, causalità. Gran parte delle iniziative industriali salta questa domanda e usa l'ontologia come middleware operativo, e regge finché i sistemi non evolvono e le scale non cominciano a interagire. La chiarezza ontologica dovrebbe precedere la formalizzazione, non seguirla come ripensamento, ed è l'esatto opposto di come si procede di solito, dove prima si costruisce il grafo, poi si calcolano gli embedding, e la semantica la si deduce alla fine.

Va conosciuto anche un prezzo tecnico, perché spiega perché molte cose qui sono più difficili di quanto sembrino. OWL, il linguaggio più diffuso per le ontologie formali, nella sua variante trattabile è progettato attorno alla decidibilità: vuole che un motore automatico concluda sempre un ragionamento in tempo finito, e per garantirlo rinuncia a un po' di espressività. Il meta-modeling diventa difficile, la semantica contestuale si indebolisce, l'auto-riferimento è vincolato. Sono il prezzo della decidibilità, non difetti, e spesso conviene pagarlo, a patto di sapere che lo si sta pagando. Cinquant'anni fa Arthur Koestler coniò la parola *holon* per indicare qualcosa che è insieme un tutto e una parte, e questa doppia natura manda in crisi la decomposizione pulita, i confini netti, le gerarchie che si contengono ordinatamente. La realtà organizzativa è fatta di *holon*, di cose che sono reparto e nodo di una rete più grande nello stesso momento, e i grafi piatti, fatti solo di nodi e archi, quella struttura la simulano per convenzione, accumulando una fragilità semantica che si scopre quando rimediarla costa caro.

Detti i limiti, resta una ragione tecnica per cui ontologia e grafo si sposano bene, utile da conoscere perché spiega perché i database relazionali arrancano proprio qui. In un grafo ogni nodo tiene fisicamente i puntatori alle sue relazioni, così attraversare un arco è un salto diretto, di costo costante, indipendente da quanto è grande il grafo. In un database relazionale la stessa domanda, chi è connesso a chi a due, tre, quattro passi di distanza, diventa una catena di giunzioni che si moltiplicano e degradano in fretta, quel disallineamento tra come pensiamo le relazioni e come le memorizza una tabella che gli informatici chiamano *impedance mismatch*. E poi c'è il ragionamento vero: su un grafo tipizzato con un'ontologia sopra, la transitività diventa automatica, se A gestisce B e B gestisce C allora A gestisce C anche se nessuno l'ha scritto, e l'ereditarietà pure, una query su "dipendente" raccoglie anche i manager se il manager è un sottotipo di dipendente. Sono risposte a domande mai formulate, e distinguono una struttura che ragiona da un archivio che si limita a restituire ciò che gli hai messo dentro.

03 La struttura come frontiera

Per vent'anni la parola d'ordine nei progetti dati è stata qualità. Dato pulito, dato affidabile, dato verificato. Giusto, e non è finito, però il vincolo che oggi blocca il valore dell'AI si è spostato di un gradino, e quel gradino quasi nessuno lo guarda. [La qualità del dato](#) dice all'organizzazione se un'informazione è affidabile, la struttura del dato dice alla macchina cosa quell'informazione significa e come le cose si relazionano tra loro. Due domande diverse, e il ragionamento vive nella seconda.

Dentro c'è una distinzione sottile e decisiva, tra metadati e struttura. I metadati descrivono cosa è un documento, e bastano per ritrovarlo. La struttura rende esplicito come le cose si relazionano, e serve per ragionarci sopra. Puoi avere un archivio perfettamente catalogato, ogni file col suo tag, e restare incapace di rispondere a una domanda semplice come qual è la nostra esposizione complessiva verso un certo paese, perché quella risposta non sta in nessun documento singolo, sta nelle relazioni tra fornitori, contratti, giurisdizioni, materie prime, e nell'archivio ben catalogato quelle relazioni non esistono da nessuna parte, se non nella testa di chi le ricostruisce a mano in due settimane.

Seguiamo quella domanda fino in fondo, perché mostra la differenza in modo fisico. I dati dei fornitori, in tabelle piatte, bastano per pagare una fattura: un identificativo, un nome, un valore di contratto, dei termini, un contatto. Non bastano per l'esposizione verso un paese, perché quella domanda vive nelle connessioni. In un grafo, ogni fornitore, ogni giurisdizione, ogni materia prima, ogni contratto, ogni quadro regolatorio è un nodo tipizzato, e la domanda diventa un percorso: parti dal paese, raggiungi tutti i fornitori che vi operano, poi le materie prime e i contratti collegati, poi i fornitori dei fornitori, e leggi le concentrazioni che si formano lungo il cammino. Un percorso attraverso il grafo, generato quando serve, e non un foglio di calcolo assemblato a mano in due settimane. La stessa struttura che rende la domanda possibile la rende difendibile: sotto un recupero piatto produci la policy che si sarebbe dovuta applicare, una difesa costruita a posteriori; sotto un grafo ricostruisci la decisione specifica che è stata presa, un record. La qualità rende una risposta corretta, la struttura la rende difendibile.

Qualità e struttura

La qualità dà nodi affidabili, la struttura dà le relazioni. Serve ragionare, e difendere, solo dove ci sono entrambe.



Fig. 3 – Qualità e struttura: si ragiona, e si difende, solo dove ci sono entrambe.

Non è una moda del 2026, e lo si capisce da una storia che vale trent'anni. Nel 1998 c'era chi digitalizzava giornali regionali catalogando ogni articolo con quindici campi di uno standard di metadati, il Dublin Core, e da quei metadati nacque il primo flusso di ricavi, perché i contenuti finalmente si ritrovavano. Poi ogni articolo venne collegato a una directory di attività, il concessionario d'auto, l'agenzia immobiliare, il cacciatore di teste, e da quelle relazioni nacque il secondo flusso di ricavi. Era un knowledge graph senza il vocabolario di oggi, e la sequenza è la stessa che stiamo vivendo a scala aziendale, prima nessuna struttura, poi i metadati, poi l'ontologia di dominio, poi il grafo esteso a tutta l'impresa. Il salto di scala è enorme, la logica è identica.

Ai decisori un'immagine funziona meglio di qualsiasi schema. Il piano dei conti definisce cosa è ricavo e cosa è costo, e nessuna azienda seria lo lascia decidere per caso, lo custodisce, lo governa, lo difende. Le definizioni delle entità in un knowledge graph, cosa è un cliente, cosa è un prodotto, cosa è un fornitore, hanno oggi lo stesso peso del piano dei conti, ma su una superficie molto più larga, perché toccano tutto ciò che sta a valle: i numeri che riporti, gli obblighi in cui incappi, le persone che un sistema di AI tratterà come bersaglio di una decisione. Il chart of entities è il nuovo chart of accounts.

Da lì discende un concetto che ai leader dovrebbe far rizzare le antenne, il *verification premium*. Un'organizzazione può verificare solo ciò su cui i suoi sistemi sanno ragionare, e a scala il ragionamento è funzione della struttura. Ci sono quattro tipi di indicatore: quelli a posteriori, che dicono un fatto già accaduto e a cui basta un dato affidabile; quelli anticipatori, che leggono un segnale; quelli predittivi, che stimano una probabilità; e quelli ragionati, che derivano una conclusione da premesse. Solo l'ultimo richiede l'ontologia, e solo l'ultimo non si può falsificare aggiungendo altri dati. Puoi avere tutti i dati del mondo, ma se non puoi ragionarci sopra in modo tracciabile, non puoi dimostrare che la tua conclusione regge.

C'è una formula, ruvida e vera, che ai board arriva dritta: [non puoi scalare l'AI senza scalare la semantica](#). L'ontologia, letta così, è uno schema vivo ed eseguibile, e non un documento statico appeso a una parete: quando interroghi il sistema non stai interrogando le tabelle grezze, stai interrogando la sua visione del mondo. Si concretizza in tre elementi che diventano regole applicate al momento della scrittura, non buone intenzioni. Un sistema di tipi, le etichette sui nodi e i tipi sugli archi, presi dalle definizioni delle classi. Uno schema delle proprietà, cosa è obbligatorio e cosa facoltativo per ogni tipo, verificato quando il dato entra. E dei vincoli sulle relazioni, quali legami sono ammessi tra quali tipi, un cliente può emettere un ordine, un fornitore no, con la regola applicata all'inserimento o alla query. Il "cliente" del marketing, chi ha lasciato un'email, e il "cliente" della finanza, chi ha completato una transazione pagata, smettono di divergere, perché l'ontologia crea un'unica fonte di verità semantica interna all'azienda, il resoconto ufficiale di come quell'impresa definisce le proprie cose, e il grafo la impone a livello di storage e di interrogazione. Non è una verità universale, e più avanti ci torno, è la coerenza di un'organizzazione con se stessa, che è già molto più di quanto la gran parte delle aziende possieda oggi. La stabilità semantica diventa una proprietà del sistema, non un auspicio.

[I dati senza semantica sono rumore](#), detta dal lato di chi costruisce database a grafo. Avere petabyte sparsi in decine di sistemi, ognuno coi suoi identificatori impliciti e le sue assunzioni non dichiarate, non ti rende data-driven, ti rende un'azienda che ha molti dati e poca conoscenza. La struttura trasforma il rumore in qualcosa su cui una macchina, e prima ancora un'organizzazione, può ragionare. E quella struttura, in forma implicita, esiste già dentro l'azienda. Il grafo latente è lì, nelle teste delle persone e nelle giunture tra i sistemi. La scelta è renderlo esplicito e governarlo, oppure lasciarlo formare per conto suo.

04 Il grounding contro l'allucinazione

Arriviamo al motivo vero, quello per cui l'ontologia è tornata al centro dopo anni da tema per convegni accademici. Il motivo si chiama modello linguistico, e il modo più onesto di dirlo è che un LLM è straordinario a produrre testo plausibile e non è progettato per essere corretto. Due qualità diverse, che confondiamo di continuo perché il testo plausibile, mentre lo leggiamo, ci sembra corretto.

I modelli sono [riconoscitori di pattern](#) straordinari, senza alcuna comprensione intrinseca della verità, e va benissimo per molte cose e per niente affatto per altre. Un "probabilmente corretto" si accetta quando suggerisci un film, non nella manifattura, nella supply chain, nell'antifrode, in medicina, dove una risposta sbagliata detta con sicurezza costa denaro, conformità, a volte vite. Qui due filoni dell'intelligenza artificiale tenuti separati per decenni tornano a incontrarsi, quello simbolico fatto di regole e struttura e quello statistico fatto di apprendimento dai dati. L'ontologia porta la struttura, il modello porta la flessibilità, e insieme danno quello che nessuno dei due dà da solo, un sistema flessibile che però sa quando dice una cosa vera e quando la sta indovinando. Il grounding, l'ancoraggio del modello a una conoscenza deterministica esterna, è il ponte tra i due mondi.

La sequenza da imparare a memoria è che il modello va usato come ultimo passo, non come primo. Se lo usi per ragionare su un dominio strutturato, prodotti, pazienti, conti correnti, policy legali, stai chiedendo la cosa giusta allo strumento sbagliato. Il ragionamento va fatto prima, sulla struttura, e al modello lasci l'ultimo miglio, mettere in linguaggio una risposta già verificata. Senza ontologia hai dei dati, con l'ontologia hai un significato: l'ontologia dà il significato, il knowledge graph dà i fatti veri di adesso, e nessuno dei due, da solo, basta.

L'anti-allucinazione si gioca su un piano più concreto di quanto si creda, e si capisce con un esempio banale. La caratteristica "impermeabile" è valida per una giacca, priva di senso per un paio di cuffie. Un modello lasciato a sé, se gli passi frammenti di testo su cuffie e su impermeabilità, cuce una frase scorrevole in cui le cuffie diventano impermeabili, e te la consegna con la sicurezza con cui ti direbbe che il cielo è azzurro. Un'architettura a tre strati ferma esattamente questo. Lo strato uno è la tripla, il fatto nudo. Lo strato due è il vincolo, scritto in OWL: la proprietà "ha caratteristica" ha un dominio e un intervallo, e una tripla che li viola viene bloccata lì, prima ancora di arrivare al modello, ed è lì che vive la guardia semantica. Lo strato tre è il frame linguistico, il template deterministico che trasforma il fatto verificato in una frase leggibile. La frase che esce è risolta, non generata.

I tre strati che fermano l'allucinazione

La frase che esce è risolta, non generata. La guardia semantica blocca il fatto invalido prima che il modello lo veda.

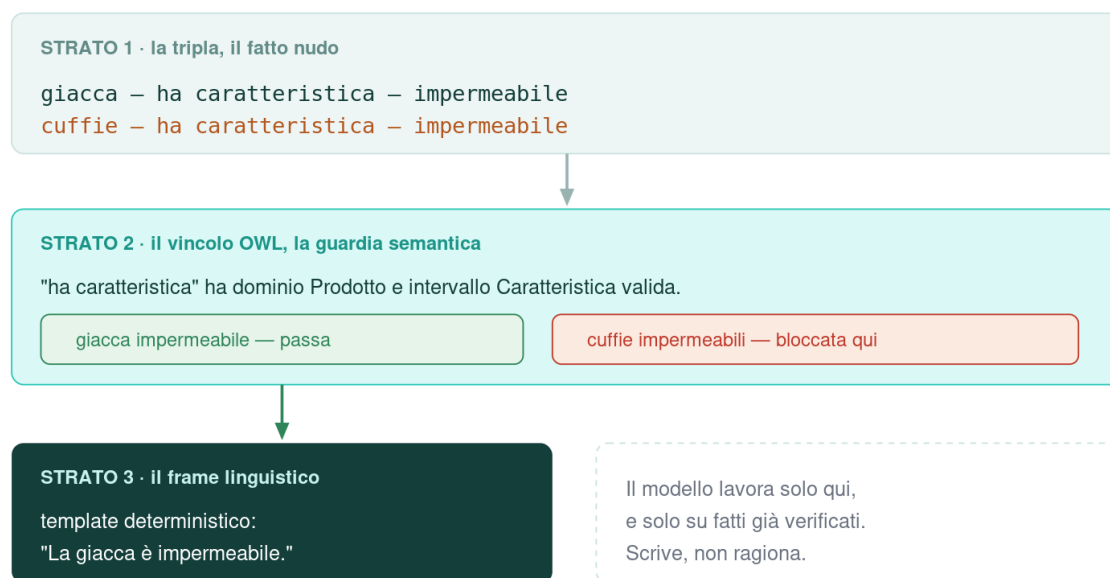


Fig. 4 – I tre strati che fermano l'allucinazione.

Lo stesso meccanismo, visto dal lato delle annotazioni: il knowledge graph è affidabile ma illeggibile per un non esperto, l'LLM è leggibile ma inaffidabile, e la mossa è usare il modello come traduttore rigidamente vincolato dalle regole dell'ontologia, mai come fonte di verità. La verità sta nel grafo, la scorrevolezza la mette il modello, e le annotazioni impediscono al modello di inventare o di lasciar cadere per strada un dato.

Un esempio quotidiano rende la cosa più concreta di ogni schema. Una ricerca su un sito di commercio: giacche impermeabili blu sotto i duecento euro, taglia M, spedizione in due giorni. Un modello solo con dei frammenti può restituire una risposta scorrevole e sbagliata, magari un prodotto esaurito, o "impermeabile" preso come un vezzo descrittivo invece che come una caratteristica binaria da verificare.

Con l'ontologia davanti, ogni pezzo della richiesta si risolve pulito: blu è un valore dell'attributo colore, impermeabile è vero o falso, taglia M è un filtro sulle varianti, due giorni è un vincolo sulla spedizione. Il grafo fornisce i fatti verificati di adesso, disponibilità reale, prezzo reale, tempi reali, non le statistiche di quando il modello è stato addestrato. E l'orchestrazione costruisce una query precisa, caratteristica impermeabile uguale vero, prezzo sotto duecento, taglia M disponibile, spedizione entro due giorni, giacenza sopra zero. La decisione interpretativa non avviene dentro il modello a runtime, avviene una volta, quando progetti l'ontologia. Lo stesso pattern scala oltre il commercio, ovunque una risposta plausibile ma sbagliata costi cara: in sanità, tra sintomi, diagnosi e controindicazioni; in finanza, tra stati di conto, regole e vincoli regolatori; nella ricerca documentale interna, tra policy, catene di approvazione e controllo degli accessi.

Il problema, del resto, ha una storia più lunga dell'attuale ondata di modelli, e conoscerla aiuta a non prenderlo come una novità assoluta. La conversione di dati strutturati in linguaggio naturale è un campo di ricerca da anni, con benchmark dedicati, come la sfida WebNLG, costruita per misurare quanto bene un sistema traduce triple RDF in prosa leggibile. I primi approcci a reti neurali erano fluenti ma lasciavano cadere dati specifici e faticavano con entità mai viste, e le annotazioni ontologiche nacquero per curare proprio quella malattia, non per rendere il testo più bello, per impedire al sistema di perdere o inventare fatti mentre lo rende leggibile.

Va detta una cosa che smonta un equivoco diffuso, perché in molte aziende la parola magica del momento è GraphRAG, pronunciata come se fosse la soluzione. Il recupero aumentato che pesca da un grafo invece che da un mucchio di testo è un passo nella direzione giusta, riduce le allucinazioni, dà al modello materiale migliore da cui partire. Ma resta una tecnica, non un'architettura. Non impone la semantica, non valida la coerenza strutturale dei frammenti che recupera, non impedisce di mescolare nodi non correlati in una risposta plausibile e sbagliata. Con GraphRAG il modello indovina meglio, con materiale più buono, però continua a indovinare. La differenza tra un sistema che indovina bene e uno che sa è tutta in dove metti la validazione, prima che il modello parli o dopo che ha già parlato.

05 Il grafo come contratto

Una cosa colpisce quando smetti di guardare i singoli prodotti e guardi la forma. [Mettendo a confronto](#) due mondi che partono da estremi opposti, lo stack aperto e componibile fatto di pezzi open source da una parte e la piattaforma verticalmente integrata di Palantir dall'altra, si scopre che convergono sulla stessa forma. Un grafo tipizzato al centro, con entità e relazioni che hanno un nome. Gli indici, quelli vettoriali e quelli lessicali, ai bordi, come porte d'ingresso per trovare il punto da cui partire. Uno strato di orchestrazione che tratta il recupero e la scrittura come strumenti a disposizione del modello. E la valutazione avvolta attorno a tutto, perché senza misurare non sai se il sistema funziona o ti sta mentendo bene.

Il cuore della questione sta in tre parole, il grafo è il contratto. Lo schema dell'ontologia lavora contemporaneamente su tre fronti. È un prompt, perché dice al modello con che entità e relazioni ha a che fare. È un validatore, perché una scrittura che viola lo schema viene rifiutata. È una superficie di API, perché definisce cosa si può chiedere e cosa si può fare. E i modelli ragionano nettamente meglio contro un grafo tipizzato con relazioni nominate che contro testo grezzo o contro una zuppa di collegamenti senza etichette.

L'architettura di riferimento

La forma su cui convergono tutti: grafo tipizzato al centro, indici ai bordi, orchestrazione sopra, valutazione attorno.

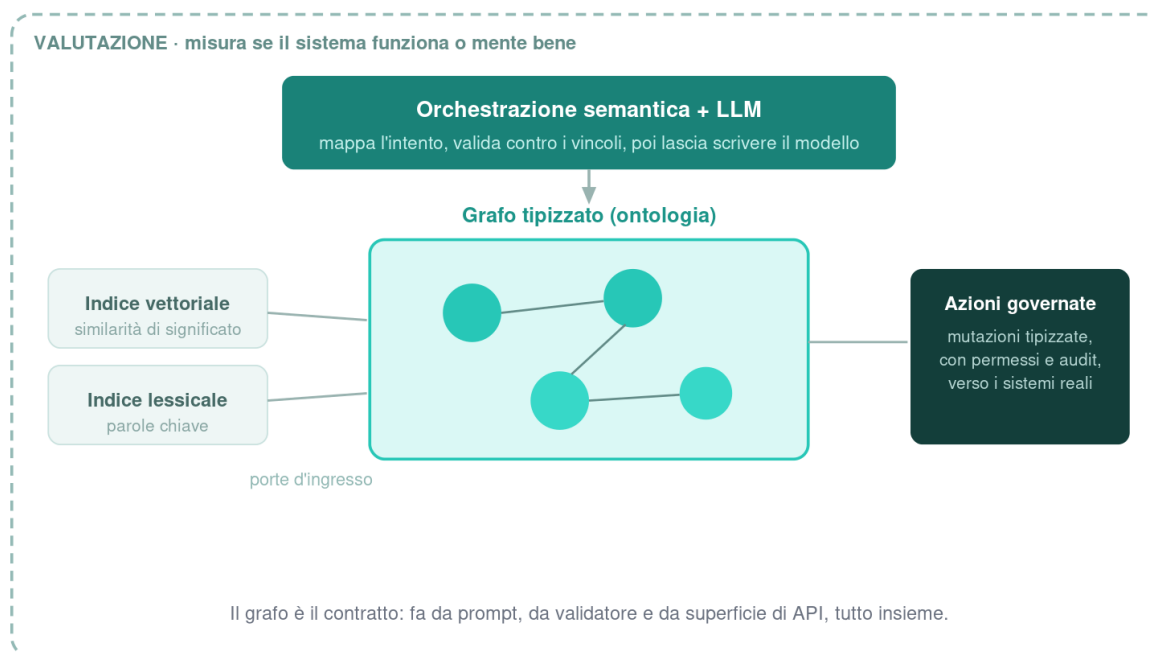


Fig. 5 – L'architettura di riferimento.

Palantir, che piaccia o no come azienda, ha portato questa idea al livello di maturità più alto in circolazione, e conviene capirla anche solo per copiarne l'impostazione. La sua Ontology ha tre primitive. Gli object type sono gli schemi delle entità, con proprietà tipizzate e una chiave primaria, alimentati dalle pipeline dei dati. I link type sono le relazioni, tipizzate e direzionali. E gli action type, la parte che quasi tutti gli stack aperti ignorano, meritano una sezione a sé e ci torno tra poco. Sopra tutto girano funzioni che eseguono il recupero dentro il modello di sicurezza, e questo significa una cosa non da poco, un utente ottiene dal grafo solo gli oggetti che ha il diritto di vedere. Il filtro dei permessi è garantito per costruzione, non appiccicato su cose già fatte. Attorno all'Ontology si impilano strati, la costruzione delle pipeline che alimentano il grafo, la logica che usa i modelli come strumenti, le applicazioni a basso codice per gli utenti finali, e tutto è attraversato da versioning, tracciabilità della provenienza e modello di sicurezza che si propaga fino al contesto del modello. La lezione è pulita: da una parte un ecosistema componibile, portabile, con costi unitari migliori, dove però la governance te la costruisci tu; dall'altra una piattaforma integrata e opinata, dove sicurezza e tracciabilità sono native ma il prezzo è un lock-in strutturale, perché un'organizzazione che ha codificato le proprie operazioni in migliaia di funzioni dentro quell'Ontology non migra con leggerezza. Nessuna delle due è giusta in assoluto, è una scelta consapevole tra due modelli di controllo, da fare sapendo cosa si guadagna e cosa si consegna.

Sotto tutto questo un dettaglio determina in silenzio il tetto di qualità del sistema, e quasi nessuno lo guarda: come spezzi i documenti in blocchi e come recuperi. La ricerca densa, per similarità di significato, e la vecchia ricerca per parole chiave falliscono in modi complementari, quindi conviene eseguirle entrambe e fondere i risultati. Sopra si mette un riordino più fine con un modello che confronta a coppie domanda e candidato. E la qualità del tutto poggia su come hai tagliato i testi in partenza, perché un chunking fatto male mette un soffitto basso a qualsiasi cosa venga dopo, per quanto sofisticato sia il modello. Il vero banco di prova di un database vettoriale è la forza del recupero filtrato dai permessi, mentre i numeri della ricerca non filtrata sono marketing.

Sopra il grafo sta il ponte, il semantic orchestration layer, il pezzo di cui si parla meno e che fa più lavoro. Riceve l'intenzione dell'utente, la mappa sui concetti dell'ontologia, sceglie le entità e le relazioni giuste, costruisce la query, valida i dati contro i vincoli prima che il modello li veda, e solo allora lascia scrivere il modello. Il mestiere del modello si riduce a scrivere, e la decisione interpretativa, quella pericolosa, si sposta al momento in cui progetti l'ontologia, una volta, fatta bene, invece di ripetersi a ogni singola risposta dentro la scatola nera.

Un livello ulteriore i CIO più attenti lo hanno già in agenda, ed è il contesto. In un mondo dove i modelli di base convergono e si somigliano sempre di più, il fossato difendibile sta nello strato di orchestrazione del dato, e le applicazioni che girano su un modello senza quello strato non hanno alcuna difendibilità. Ci sono quattro flussi di contesto da orchestrare in un unico payload, l'intento dell'utente nella sessione corrente, la sua storia di lungo periodo, il contesto di sistema coi suoi guardrail, e i dati esterni in tempo reale. La sfida ingegneristica è decidere cosa merita un posto nella finestra di contesto e cosa va scartato, prima ancora che concatenare le fonti, perché ogni token ha un costo diretto e a scala quel costo si mangia i margini. Su storie lunghe mesi o anni la soluzione è una struttura di memoria a grafo, che decide in tempo reale cosa far emergere, cosa comprimere, cosa dimenticare del tutto.

I quattro flussi di contesto

Ogni token ha un costo. L'orchestrazione decide cosa far emergere, cosa comprimere, cosa dimenticare.

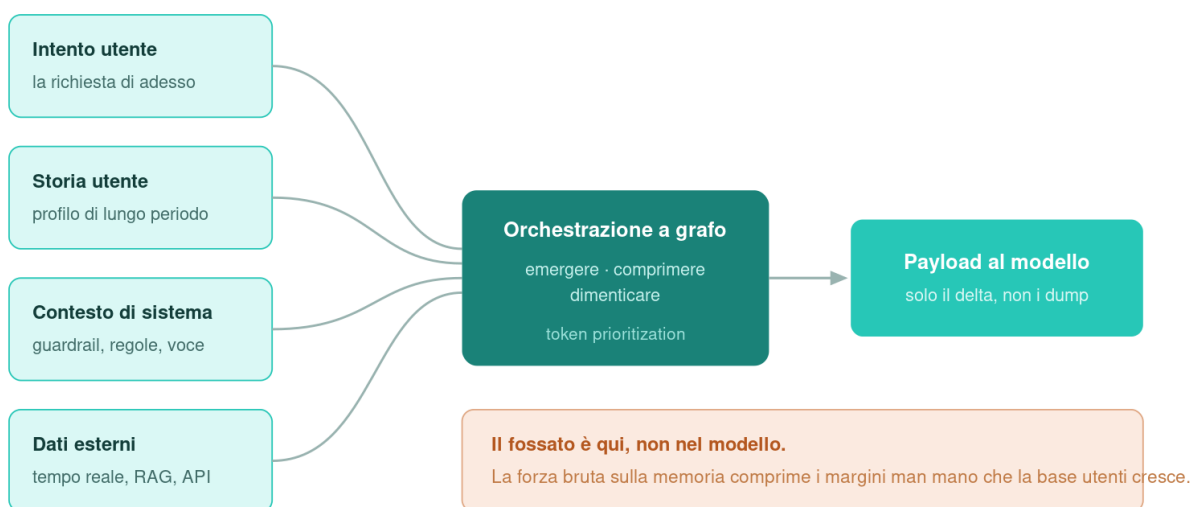


Fig. 6 – I quattro flussi di contesto e la token prioritization.

Due fallimenti reali valgono più di qualsiasi principio, perché mostrano dove finiscono i soldi. Un assistente per il customer success in un'azienda di software, con un approccio ingenuo, tirava cinquantamila parole di email grezze dentro la finestra di contesto, saturava il limite, tagliava proprio le email più recenti e critiche, e mancava un rinnovo contrattuale, per aver dato priorità al volume invece che alla pertinenza. La versione ingegnerizzata passava al modello solo i riassunti densi dei log vecchi più il testo integrale delle tre interazioni più recenti. Un robo-advisor finanziario, a ogni aggiornamento di pagina, ripassava l'intera storia del portafoglio, venti titoli, cinque notizie, con costi di API insostenibili perché il novanta per cento di quei token era dato statico e ridondante; la correzione era architetturale, mettere in cache ciò che non cambia e iniettare solo il delta, i valori effettivamente cambiati oltre una soglia. Ogni token in ingresso e in uscita ha un costo diretto, e l'approccio a forza bruta sulla memoria

comprime i margini man mano che la base utenti cresce. La tua architettura del contesto impatta direttamente il conto economico, ed è una frase che un CFO capisce meglio di qualsiasi schema.

L'ultimo strato è quello che trasforma un modello in un agente. Lo riassume bene [un'equazione](#) che si tiene a mente, AI uguale LLM più harness. L'harness è tutto ciò che sta attorno al modello, il ciclo, gli strumenti, la memoria, i guardrail, i tentativi ripetuti, la tracciabilità. Il modello genera, l'harness decide se fidarsi. Il modello è la parte piccola, lo cambi il mese prossimo con uno migliore, mentre l'harness è quello che costruisci e mantieni davvero, ed è lì che si gioca l'affidabilità. Un motore di ricerca sa dirti qual è la regola adesso, un harness sa dirti cosa è cambiato dall'ultima volta, e in azienda la seconda domanda vale molto più della prima.

Dentro un harness lo sforzo ingegneristico va speso quasi tutto fuori dal modello. Il ciclo è osserva, ragiona, agisci: cerca fonti ufficiali aggiornate, classificale ed estrai le regole con numeri e date, scrivi un output usabile da un umano. Dopo aver ragionato l'agente si autocontrolla, ha abbastanza fonti, sono abbastanza affidabili, e se non basta riscrive la query verso fonti migliori e ripete, fino a un tetto di tentativi. Attorno al ciclo vivono gli strumenti di ricerca intercambiabili con fallback automatico, la validazione dello schema JSON che ritenta con un prompt correttivo se l'output è troncato, la memoria che confronta il nuovo risultato con l'ultimo salvato e mostra cosa è cambiato senza creare duplicati. Sempre lo stesso principio, a ogni livello di questo discorso: il valore durevole vive nel sistema che decide quando fidarsi del componente intelligente, più che nel componente, e quel sistema lo costruisci e lo mantieni tu, mentre il modello sotto puoi cambiarlo domani.

06 Le azioni governate

Riprendo gli action type di Palantir, perché è qui che la conversazione su ontologie e AI esce dalla ricerca documentale ed entra nelle operazioni vere. Quasi tutti i sistemi RAG in circolazione fanno recuperare informazione. Poi, quando serve scrivere qualcosa, cambiare uno stato, far succedere qualcosa nel mondo, si affidano a scorciatoie improvvisate, chiamate dirette al database, script cuciti a mano. Le azioni sono la frontiera che il recupero puro ignora.

Un action type è una mutazione dichiarata, tipizzata, con permessi verificati, regole di validazione, effetti collaterali espliciti, una traccia di audit, e la possibilità di richiedere l'approvazione di un umano prima di eseguirsi. L'ontologia diventa leggibile e scrivibile, ma solo attraverso azioni, mai attraverso una maniglia diretta sul database. Il principio lo vorrei scritto in ogni progetto: definisci un vocabolario chiuso di mutazioni possibili, con le loro precondizioni, e dai al modello quello, mai un accesso libero a scrivere ciò che vuole.

Dal lato dell'harness la stessa idea si veste da governance operativa. I guardrail non sono decorativi, possono davvero fallire e fermare il processo. Un controllo in ingresso blocca richieste sospette prima di qualsiasi chiamata di rete. Un controllo in uscita verifica che l'output porti con sé una fonte, un livello di confidenza, un disclaimer, e segnala come allarme, non come via libera silenziosa, l'assenza di fonti ufficiali. Un episodio vale più di una teoria: su un tema di regole per i visti un agente aveva ottenuto otto fonti, nessuna ufficiale, il controllo è correttamente diventato giallo, l'agente ha riscritto da solo la ricerca verso fonti governative e ne ha trovate sette su otto ufficiali. In quel momento ha smesso di sembrare uno script.

Il grounding contro le allucinazioni incontra qui la governance e l'umano nel ciclo, sul terreno delle azioni, e non è un caso. Recuperare un'informazione sbagliata costa una risposta imprecisa. Eseguire un'azione

sbagliata su un asset reale costa molto di più, e la differenza è esattamente lo strato che decidi di costruire tra il modello e il mondo. Un'ontologia che descrive solo com'è fatto il dominio è metà del lavoro. L'altra metà descrive cosa è lecito farci, da chi, con quali controlli, e lascia traccia di ogni cosa fatta.

Cambia così anche la natura dell'ontologia, da fotografia a strumento operativo. Nel modello di Palantir il grafo è scrivibile oltre che leggibile, e le scritture possono tornare fino ai sistemi sorgente, ma sempre e solo passando dalle azioni, mai con una scrittura diretta e opaca. Ogni modifica porta con sé chi l'ha fatta, quando, con quale validazione, e resta ricostruibile grazie alla tracciabilità della provenienza. È la differenza tra un'AI che ti dice cosa dovresti fare e un'AI che fa, dentro un recinto di regole che qualcuno ha scritto e di cui qualcuno risponde. Più le aziende collegano agenti a questi grafi, più il recinto conta, perché un agente che può solo leggere sbaglia una risposta, un agente che può agire senza recinto sbaglia un'operazione, e la seconda ha un costo che si misura in denaro vero, non in imbarazzo.

07 La frode come rete

Fin qui il ragionamento è stato architetturale, e rischia di restare astratto senza un problema che un CEO riconosce al volo. La rilevazione delle frodi è quello giusto, perché mostra tutto insieme, perché la struttura conta, perché la semantica conta, e perché la velocità di esecuzione non basta senza il significato.

Il punto di partenza ribalta il modo classico di guardare il problema. La frode moderna non è un problema di transazione, è un problema di rete. Una singola operazione, guardata da sola, sembra legittima quasi sempre. È il pattern di relazioni attorno a quella operazione a raccontare la storia vera: decine di account collegati allo stesso dispositivo, allo stesso numero, alla stessa persona giuridica, in configurazioni che nessun controllo transazione per transazione può vedere. Un grafo che si limita a memorizzare che il dispositivo A è collegato all'account B non basta, perché non distingue una famiglia che condivide un portatile da una rete di frode che usa un solo dispositivo per centinaia di account falsi. Serve la semantica, sapere cosa significa quel nodo, cosa significa quell'arco, quali vincoli valgono, come il rischio si propaga tra tipi di entità diversi.

Qui l'ontologia entra in modo tangibile. Definisce i tipi di entità, l'entità legale, il rappresentante autorizzato, il dispositivo, l'indirizzo IP, la transazione, il conto. Definisce le relazioni, controlla, è registrato presso, è associato a, ha origine da. Definisce le proprietà, il codice di giurisdizione, l'impronta del dispositivo, il punteggio di rischio, l'orario. E definisce i vincoli, la regola di sospetto, un dispositivo associato a più di cinque entità legali distinte in ventiquattro ore va in revisione. Quella regola vive nell'ontologia, scritta una volta, non cablata dentro il codice di venti pipeline diverse. Gli umani definiscono l'ontologia, il database a grafo la esegue in automatico: il design è lavoro umano di settimane, l'esecuzione è a tempo di macchina, senza intervento, su ogni pagamento, in millisecondi, prima che la transazione si completi.

Il flusso a runtime, per un pagamento cross-border, si legge come tre controlli concatenati. Prima il dispositivo, è associato ad altri conti e questi che punteggio di rischio hanno. Poi l'entità legale, la giurisdizione è sanzionata, il rappresentante è su una watchlist. Poi il pattern della transazione, la combinazione di dispositivo, IP, entità e importo somiglia a una firma di frode già vista. Tre passaggi, un grafo semantico che li attraversa sapendo cosa attraversa, una risposta prima che i soldi si muovano. I numeri che accompagnano questi sistemi, quando funzionano, sono importanti: una banca globale che riduce i falsi positivi di oltre il quaranta per cento nella sorveglianza degli scambi, un operatore di credito

che passa da query oltre i tre secondi, con il trenta per cento delle domande ad alto rischio mancate, a meno di otto millisecondi dopo aver ricostruito il tutto su un grafo. Una precisazione onesta, però, la devo: cifre come queste arrivano quasi sempre da chi quelle piattaforme le vende, senza baseline pubblica né il lato dei costi, e vanno lette come indicazioni di direzione, non come prove. L'argomento di questo articolo non poggia su di loro, poggia sulla logica del grafo, e i numeri servono solo a dare un ordine di grandezza a ciò che quella logica rende possibile.

Il valore ulteriore è organizzativo prima che tecnico. In un'azienda che gestisce rischio, il team KYC, chi monitora le transazioni e chi indaga sulle frodi guardano gli stessi dati e definiscono "rischio" in modo leggermente diverso ciascuno. L'ontologia diventa il livello semantico condiviso che allinea l'intera organizzazione su una definizione sola, e la rilevazione delle frodi passa dalla caccia reattiva alla regola che ha fatto scattare l'allarme a un'intelligenza proattiva sulle relazioni. Senza ontologia ogni investigatore ricuce a mano i pezzi. Con l'ontologia la ricucitura la fa il sistema, e la fa sempre allo stesso modo.

08 Lo standard e l'estensione

Arriva la domanda pratica, subito dopo aver capito perché serve: la costruiamo noi, questa ontologia. La prima risposta è controintuitiva per chi ha una cultura ingegneristica del fare in casa. In molti settori la parte più difficile l'ha già fatta qualcun altro, spesso decenni fa, con anni di lavoro di comitato, governance formale e mappature verso ogni altro standard del settore.

[Ricostruire da zero](#) concetti come "paziente" o "diagnosi" è rilavoro travestito da artigianato, e gli schemi puliti fatti in casa sono il modo con cui finisci per reinventare in privato SNOMED, un'etichetta di nodo alla volta. Conviene sapere cosa c'è già sullo scaffale, perché è tanto e quasi nessuno in azienda lo conosce.

In sanità e farmaceutica lo scaffale è affollato. SNOMED CT è la terminologia clinica, centinaia di migliaia di concetti, usata in oltre cinquanta paesi. ICD-11 è la classificazione delle malattie dell'Organizzazione Mondiale della Sanità. MedDRA copre gli eventi avversi ed è di proprietà dei regolatori stessi. RxNorm normalizza i nomi dei farmaci. UMLS fa da stele di Rosetta e mappa gli uni sugli altri. La Gene Ontology descrive la funzione dei geni con oltre un milione di annotazioni sostenute da evidenza sperimentale. In finanza c'è FIBO, la Financial Industry Business Ontology, costruita da EDM Council e OMG, con migliaia di classi in produzione. Nella supply chain non c'è un'ontologia unica ma uno stack, GS1 con EPCIS per gli eventi, SCOR per i processi, lo standard W3C PROV per la provenienza. Nel legale c'è LKIF. E come collante generale ci sono schema.org, che usi ogni volta che ottimizzi un sito per i motori di ricerca senza saperlo, e Wikidata.

Settore	Standard	Cosa copre	Chi lo mantiene
Sanita e pharma	SNOMED CT, ICD-11, MedDRA, RxNorm, UMLS, Gene Ontology	terminologia clinica, malattie, eventi avversi, farmaci, geni	SNOMED Intl, OMS, NLM, accademia
Finanza	FIBO	entita, strumenti, securities, derivati	EDM Council, OMG
Supply chain	GS1 EPCIS, SCOR, W3C PROV	eventi, processi, provenienza	GS1, ASCM, W3C
Legale	LKIF	obblighi, permessi, proibizioni	consorzio accademico
Generale	schema.org, Wikidata	persone, organizzazioni, luoghi, prodotti	consorzio web, community

Tabella 1 – Ontologie standard per settore.

FIBO, in finanza, conta migliaia di classi in produzione, dai fondamenti alle entità di business agli strumenti finanziari ai derivati, e proprio questa ricchezza fa scattare un errore che vedo fare spesso, scambiare la dimensione per un motivo per costruire in casa. Non lo è. Quaranta classi curate, mantenute davvero, allineate ai tuoi codici, battono duemila classi scaricate e mai riviste, perché la dimensione conta pochissimo, e ciò che dà valore è che il pezzo preso in prestito è già d'accordo con quello di tutti gli altri. In sanità è ancora più chiaro: UMLS mappa tra loro SNOMED, ICD, RxNorm, MedDRA, e mappe come quelle esistono proprio perché i concetti di due standard non si allineano mai uno a uno.

La cosa che nessuno di questi standard ti dà gratis, e che è il vero lavoro, è la cura. Riusare non è gratis, scambia lavoro di modellazione con lavoro di comprensione, e di solito conviene. Le trappole sono tre. La prima, non importi tutto lo standard, prendi la fetta che serve e la mantieni, perché uno slice curato batte l'import completo solo se lo mantieni davvero. La seconda, il mapping è dove si nasconde il lavoro vero, perché i concetti di due standard non si allineano quasi mai uno a uno, e una mappatura sottilmente sbagliata è peggio di nessuna mappatura. La terza, il fit non è mai perfetto, perché lo standard incorpora la visione di qualcun altro, ed estenderlo senza romperlo richiede più comprensione, non meno.

C'è una ragione profonda per cui questi standard sono sicuri su cui costruire, ed è chi li mantiene. Non escono da un vendor, li tengono in piedi l'Organizzazione Mondiale della Sanità, i consorzi di regolatori, le biblioteche nazionali di medicina, gli enti no-profit e i consorzi accademici su fondi pubblici. È infrastruttura lenta, governata, noiosa nel senso migliore del termine, che sopravvive a qualsiasi roadmap di prodotto e a qualsiasi acquisizione. E c'è una scommessa che riguarda il futuro prossimo: man mano che colleghi agenti ai grafi, il valore di un vocabolario condiviso e leggibile dalle macchine aumenta, non diminuisce. Un agente che ragiona su dati fondati su uno standard come FIBO poggia su migliaia di classi di accordo consolidato sul significato delle parole. Un agente che ragiona su uno schema inventato lo sprint scorso poggia sull'intuizione del momento, e quando sbaglia non hai nulla a cui appellarti. L'ontologia è ciò che impedisce a un knowledge graph di scivolare in un grande mucchio di rumore duplicato con degli archi.

Il criterio strategico che ne esce lo tengo come bussola: lo standard ti rende interoperabile, le tue estensioni ti rendono interessante. Metà del valore di un knowledge graph aziendale può stare proprio nella parte che nessuno ha standardizzato, quella che nemmeno i competitor hanno modellato, perché è lì che vive la conoscenza specifica della tua impresa. Riusi dove il dominio è stabile e regolato, costruisci dove sei davvero originale, e in ogni momento sai su quale metà dei due piedi stai poggiando.

Riusa, costruisci, o ibrido

Lo standard ti rende interoperabile, le tue estensioni ti rendono interessante. Quasi sempre la risposta è ibrida.



Fig. 7 – Riusa, costruisci o ibrido.

Resta il problema di come si comincia, più psicologico che tecnico. Lo chiamano [il problema della tela bianca](#): troppi modi validi di modellare un dominio e nessun punto d'ancoraggio, così il dibattito su quali siano le nostre entità gira in tondo per giorni. Un'ancora si chiama POLE+O. POLE è un modello storico dell'analisi di intelligence e delle forze dell'ordine, quattro tipi con cui da decenni si connettono dati eterogenei tra agenzie diverse, Person, Organization, Location, Event. Il più uno è Object, il quinto tipo, il contenitore per tutto ciò che non rientra nei primi quattro: se un'entità non è persona, organizzazione, luogo o evento, è un Object. Cinque tipi base con cui partire subito. La potenza non sta nei cinque tipi, sta nel fatto che sopra il tipo base ci metti il tipo di dominio, un paziente è Person più Patient, uno sprint è Event più Sprint. Interroghi al livello base e prendi insieme diagnosi, incidenti e transazioni, e unifichi in automatico sorgenti diverse perché una Person dell'HR e una del gestionale sono già nello stesso contenitore. Un grafo che puoi interrogare in settimana uno ti insegna sul tuo dominio più di uno schema su cui hai dibattuto per un mese. Con l'onestà, dichiarata, che POLE è un punto di partenza e non un'ontologia completa, e che nei domini astratti, gli strumenti finanziari, le regole di business, l'Object che tiene tutto rischia di non tenere niente.

POLE+O, cinque tipi per partire subito

POLE (Person, Organization, Location, Event) più O (Object). Sopra il tipo base metti il tipo di dominio.



Fig. 8 – POLE+O: quattro tipi base più Object.

09 Estrazione e popolamento

Il calcolo economico di questi progetti è cambiato, perché costruire un grafo, la parte che fino a ieri costava mesi e specialisti, è diventato a buon mercato. Tecniche recenti hanno tagliato i costi di costruzione di ordini di grandezza, e in alcuni benchmark la precisione con cui un modello estrae entità e relazioni da un testo si avvicina al novantanove per cento quando gli dai uno schema di riferimento a cui attenersi, contro numeri molto più bassi quando lo lasci a briglia sciolta.

Su quel numero, però, va tenuta la testa fredda, perché una precisione anche altissima non dice quanto pesa [la parte che manca](#), e quella parte raccoglie quasi sempre le distinzioni che contano di più, i casi limite, le eccezioni, i confini di responsabilità, non il rumore distribuito a caso che verrebbe da immaginare. Un'ontologia estratta al novanta o al novantacinque per cento va benissimo per la ricerca, per la BI, per un RAG, e diventa fragile dove il margine di errore non è ammesso, la compliance, la sicurezza, la progettazione, la configurazione di prodotto. Il problema vero non sta nell'accuratezza, sta nell'allineamento del significato tra tutti quelli che quel modello lo useranno.

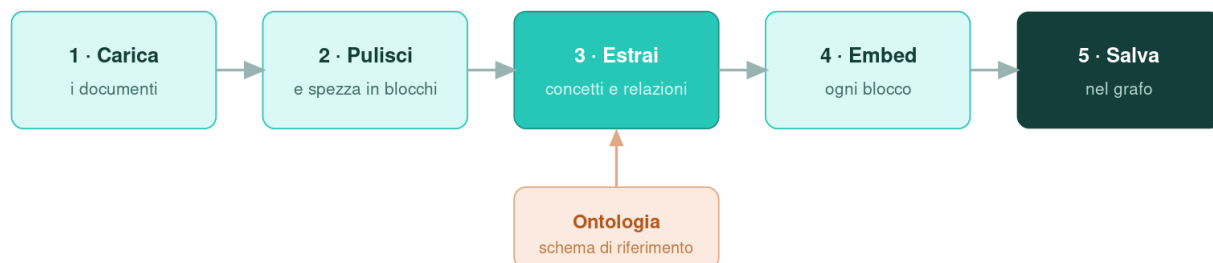
Quel dettaglio, lo schema che guida l'estrazione, è la chiave di volta e torna in ogni sezione di questo discorso, è la stessa fetta di standard curato di prima, usata qui come binario per l'estrazione.

[Una pipeline end-to-end](#) si costruisce interamente con strumenti open source, gira su un portatile e non richiede licenze. La sequenza è pulita: carichi i documenti, li pulisci e li spezzi in blocchi, estrai da ogni blocco un piccolo grafo di concetti passando all'estrattore l'ontologia come progetto opzionale, calcoli gli embedding, salvi tutto su un database a grafo collegando ogni blocco al documento e al sottografo estratto. Il pezzo che tengo da parte riguarda le query, perché ce ne sono cinque modi diversi per lo stesso grafo, dalla ricerca per similarità arricchita alla generazione automatica di query strutturate, e la lezione è che non esiste la strategia giusta, si sceglie per tipo di domanda, bilanciando costo, latenza e qualità. Più l'ontologia è precisa e descrittiva, più è facile per il modello scrivere la query giusta, e la

governance dello schema migliora direttamente l'accuratezza self-service.

La pipeline di costruzione con i modelli

Dai documenti al grafo interrogabile. L'ontologia passata come progetto guida l'estrazione e alza la precisione.



Con lo schema che guida l'estrazione la precisione arriva vicino al 99%. Senza, il modello va a naso.

Fig. 9 – La pipeline di costruzione con i modelli.

Quelle strategie, da vicino, mostrano che non c'è una via unica. Il recupero per similarità arricchito pesca i blocchi più vicini alla domanda e li completa coi contigui, buono quando serve dettaglio e citazione della fonte. I riassunti di comunità raggruppano il grafo in gruppi di nodi affini e riassumono ogni gruppo, buoni per domande di sintesi che attraversano molte fonti. La generazione diretta di query strutturate passa al modello lo schema del grafo e lo lascia scrivere l'interrogazione, ottima per domande con risposta precisa quando lo schema è ben definito. E poi le combinazioni, con un dettaglio che conta in produzione, il fallback: se la generazione della query fallisce, il sistema ripiega sul recupero per similarità invece di restituire un errore. La lezione è di portafoglio più che di tecnologia, si sceglie per tipo di domanda pesando insieme il costo in token, la latenza, la qualità, perché una risposta perfetta che costa dieci volte tanto e arriva in dieci secondi, a scala, non è sostenibile.

Sul come estrarre il dibattito è vero, e [mapparlo con onestà](#) aiuta. Le opzioni sono un modello grande usato così com'è, un modello più piccolo addestrato sul tuo dominio, una pipeline classica basata su regole, o una combinazione. Il modello grande parte subito ma costa e varia. Il modello piccolo addestrato costa all'inizio e poi è economico e costante. Le regole classiche sono deterministiche e verificabili ma fragili fuori dominio. Nessun vincitore assoluto, una decisione di architettura che dipende dai tuoi vincoli, e non un dettaglio tecnico da delegare, perché il come estrai plasma qualità, coerenza e utilità dell'intero grafo.

Sotto la scelta c'è un asse da rendere esplicito, tra approccio dall'alto e dal basso. Dall'alto uno schema o un'ontologia guida l'estrazione e i dati vengono piegati a quella struttura, ed è esattamente lo schema guidance che porta la precisione vicino al novantanove per cento. Dal basso lo schema emerge dai dati. Le pipeline classiche basate su regole hanno una dignità che l'entusiasmo per i modelli fa dimenticare, i pattern linguistici che riconoscono relazioni di tipo "è un" esistono da decenni, sono deterministici e verificabili, e in un dominio stabile lavorano senza sorprese. Gli approcci più promettenti oggi sono ibridi, dove regole ontologiche esplicite vincolano un modello addestrato a esserne consapevole, così che il ragionamento resti ancorato ai vincoli simbolici invece di allontanarsene a piacere. La regola che ne ricavo, e che lega questa sezione a tutte le altre, è che dare al modello uno schema di riferimento è il moltiplicatore che alza la qualità in ogni fase, dall'estrazione alla query.

Un caso mi ha colpito per la sua eleganza pratica. [Uno strumento](#) prende un sito o una cartella di documenti e ne induce l'ontologia dai documenti stessi, senza schema fissato in partenza, assegnando a ogni attributo e a ogni relazione un punteggio di confidenza. La parte interessante è che lo schema si auto-corregge: trovati dei nodi orfani, lo strumento chiede al modello se il problema è lo schema, il modello propone nuove proprietà, la pipeline le valida e rifà l'estrazione con lo schema allargato. Il grafo ha fatto crescere il proprio schema per adattarsi ai dati, una volta, poi si è assestato. E ogni affermazione resta tracciabile fino alla pagina di origine, con un'impronta che dice esattamente cosa è cambiato tra un crawl e l'altro. La differenza tra aver raschiato un sito una volta e avere uno snapshot verificabile. Il grafo è l'artefatto, la risposta a una domanda è solo la prima cosa che gli ho chiesto.

I numeri di quel caso danno la misura. Da un corpus di pagine di un sito di news è uscito un grafo di 779 nodi e 1131 archi, con i tipi indotti dai testi, articolo, editore, azienda, argomento, persona, borsa, indice, e le relazioni tra loro, pubblicato da, tratta di, menziona, quotato su. La confidenza è differenziata in modo onesto, il ticker Nvidia estratto da un inciso di passaggio pesa 0,6, l'industria dei semiconduttori asserita su molte fonti pesa 0,8. E una domanda come cosa è di tendenza si traduce in un'operazione concreta e verificabile, contare gli archi "tratta di" per argomento, con risultati ispezionabili uno per uno, ottantasette articoli sull'AI, settanta sul quantum, e così via. L'opposto della risposta plausibile presa sulla fiducia: ogni riga risale a un file, ogni file a una pagina con la sua impronta.

Da qui nasce anche la memoria strutturata, che vale per chi costruisce agenti. Si organizza su tre livelli sopra il modello a cinque tipi, una memoria di breve termine per la sessione, una di lungo termine fatta di relazioni persistenti tra entità, e una memoria di ragionamento che registra le tracce delle decisioni, il pensiero, l'azione, l'osservazione, come nodi collegati. Una catena di ragionamento non la attraversi in un archivio di vettori, la attraversi in un grafo. È la differenza tra un agente che ricorda cosa ha deciso e perché, e uno che riparte ogni volta da zero fingendo memoria.

10 La semantica fuori dal grafo

Adesso la controcorrente, la sezione più utile per chi tiene i cordoni della borsa perché fa risparmiare soldi e tempo, e che mi serve anche per sciogliere un equivoco che potrei aver alimentato io stesso fin qui, perché quando dico knowledge graph intendo uno strato semantico strutturato e governato di cui il grafo tipizzato è la sede più comune ma non l'unica, dato che la stessa ontologia può vivere dentro un database a grafo, dentro un motore di regole esterno, oppure in annotazioni che vincolano il modello, e a contare è il contratto semantico, molto meno il contenitore in cui lo scrivi. In ogni riunione, intanto, gira un ragionamento automatico che suona così, ci serve un'ontologia, le ontologie si scrivono in RDF, RDF è un grafo, quindi compriamo un database a grafo, e [ogni singolo anello](#) di quella catena è più debole di quanto sembri.

I database a grafo non sono né nuovi né unici. Il modello a rete trattava le relazioni come cittadini di prima classe già alla fine degli anni Sessanta, e anche i database relazionali, per definizione, sono relazionali. La forza vera dei database a grafo è stretta e reale, le query ricorsive, gli attraversamenti a profondità ignota, quando devi risalire una catena di fornitori dei fornitori o un percorso di rete di lunghezza imprevedibile. Fuori da lì il vantaggio si assottiglia in fretta.

Una distinzione chiarisce l'equivoco: quando diciamo che le relazioni contano, stiamo dicendo tre cose diverse. Può contare l'esistenza, il fatto stesso che A sia connesso a B è già informazione, e in certi domini, l'intelligence per esempio, è il cuore del lavoro. Può contare l'attraversamento, riesco ad arrivare

da A a C. E può contare la semantica, cosa significa quell'arco, è transitivo, è simmetrico, è funzionale. I database a grafo gestiscono benissimo l'esistenza e l'attraversamento, non gestiscono la semantica, che è dominio della logica e non della teoria dei grafi. Confondere i tre significati è il modo con cui si compra uno strumento per un problema che non ha.

Ci sono poi limiti pratici che chi vende racconta di rado. La variante più espressiva di OWL è indecidibile e quasi nessuno la implementa fuori dall'accademia, quella trattabile è costosa, e i database a grafo generalisti non fanno inferenza da soli, servono motori di ragionamento esterni. In ambito farmaceutico SNOMED CT usa un database a grafo come deposito, ma la classificazione logica va eseguita fuori dal database, in batch, con un reasoner dedicato, il che complica la pipeline. Le relazioni sono limitate a due estremi per volta, e appena ne servono di più, un evento con tempo, luogo, attori e importo, devi ricorrere a ipergrafi o a giri di reificazione che appesantiscono il modello. E il modello di sicurezza è spesso sottosviluppato, un percorso da A a C che passa per D può rivelare informazione che l'utente non dovrebbe vedere, e gestire questa non-monotonicità dipendente dal ruolo è tutt'altro che banale. Niente di tutto questo dice che i grafi siano inutili, dice solo che vanno scelti per il problema giusto, con gli occhi aperti sul lavoro che resta fuori dal database.

La distinzione che chiarisce tutto è tra fatti e regole. Nella logica descrittiva si parla di A-Box, le asserzioni, i fatti, ciò che un database a grafo memorizza benissimo, e di T-Box, la terminologia e le regole, l'inferenza, ciò che un database a grafo non fa. I linguaggi di interrogazione dei grafi sono ottimi ad attraversare, non ragionano, e per l'inferenza servono motori esterni. Una frase vale un'intera consulenza: un database a grafo senza un'ontologia è solo un database relazionale meno comodo. A rendere intelligente il grafo è la semantica che ci mette sopra, mai il grafo in sé, e quella semantica vive nella logica, non nella teoria dei grafi.

La proposta pragmatica è un percorso a due passi che parte da dove l'azienda già è. Passo uno, non costruire un'ontologia, esponi ai modelli le regole che hai già, perché quasi ogni impresa strutturata ha motori di regole, policy codificate, logica di business già scritta in sistemi che nessuno guarda più, e renderli accessibili a un modello tramite un protocollo standard come MCP è spesso il primo passo giusto. Passo due, se serve di più, il logic programming, con linguaggi come Datalog che garantiscono decidibilità, sono veloci, nativi sui fatti e sulle regole, e per giunta leggibili dagli stakeholder di business, cosa che di RDF e OWL non si può dire. La morale è semplice, e i database a grafo restano utili, semplicemente non devono essere la scelta di default, e di sicuro non la prima. La domanda giusta è dove metto la semantica, non se comprare o meno un grafo: dentro il grafo, in un reasoner esterno, o in annotazioni che vincolano il modello. Sono tre sedi della stessa cosa, e tutto quello che ho scritto sull'ontologia come contratto vale identico in ognuna, cambia solo dove il contratto è scritto e chi lo fa rispettare. Rispondere bene a questa domanda, prima di firmare qualsiasi licenza, è metà del risparmio.

Dove vive la semantica

Lo stesso contratto semantico può stare in tre sedi. Cambia dove è scritto e chi lo fa rispettare, non la sua natura.



Fig. 10 – Dove vive la semantica.

11 La deriva dell'ontologia

Fin qui abbiamo parlato di costruire. Quasi nessuno parla di cosa succede il giorno dopo, quando l'ontologia che governa il grafo comincia lentamente a non corrispondere più alla realtà. La chiamano [ontology drift](#), e l'osservazione di partenza dovrebbe far riflettere ogni sponsor di questi progetti. La tecnologia è matura, i costi di costruzione sono crollati, eppure l'adozione in produzione è piatta: secondo un sondaggio di settore poco più di un quarto dei knowledge graph aziendali arriva davvero in produzione, un anno dopo l'altro, quasi senza spostarsi. Se costruire è diventato facile e i grafi in produzione non aumentano, il collo di bottiglia non è più costruire. È mantenere allineato il significato mentre il business cambia.

I modelli semantici non falliscono di colpo, divergono. Una definizione cambia in un reparto, la finanza ridefinisce cosa conta come contratto attivo escludendo i rinnovi anticipati, e nessuno aggiorna il modello. Uno schema evolve per conto suo, una tabella nuova, una colonna rinominata, una pipeline rifattorizzata, e l'ontologia non ne sa niente. La stessa logica viene duplicata in cinque punti, una vista SQL, il livello di BI, il codice applicativo, il prompt dell'AI, e la si aggiorna in tre di quei cinque. Nessun sistema possiede davvero il significato, la logica si sparpaglia, non c'è più una fonte autorevole, e i team congelano le definizioni per paura di rompere qualcosa a valle. Il debito semantico si accumula in silenzio, come il [debito cognitivo](#) di cui parlo da tempo, il costo nascosto di aver delegato senza struttura, che non vedi mentre cresce e che ti presenta il conto quando meno te lo aspetti.

Un dettaglio rende il tema urgente proprio adesso, e riguarda la differenza tra la vecchia BI e la nuova AI. Una dashboard con definizioni un po' datate zoppica, mostra un trend leggermente storto, e l'analista umano che la legge se ne accorge, aggiusta, corregge a naso. Un agente no. Tratta l'ontologia come verità di riferimento e restituisce con sicurezza una risposta costruita su una logica ormai superata, e nessuno se ne accorge finché il danno non è fatto. La divergenza semantica uccide l'adozione dell'AI più in fretta di quanto uccida la BI, perché l'AI non ha il senso critico dell'analista che compensa a valle. Fallisce in silenzio, e il silenzio è la parte pericolosa.

La deriva dell'ontologia

Il modello semantico non fallisce di colpo, diverge. E l'AI, a differenza della BI, sbaglia in silenzio.

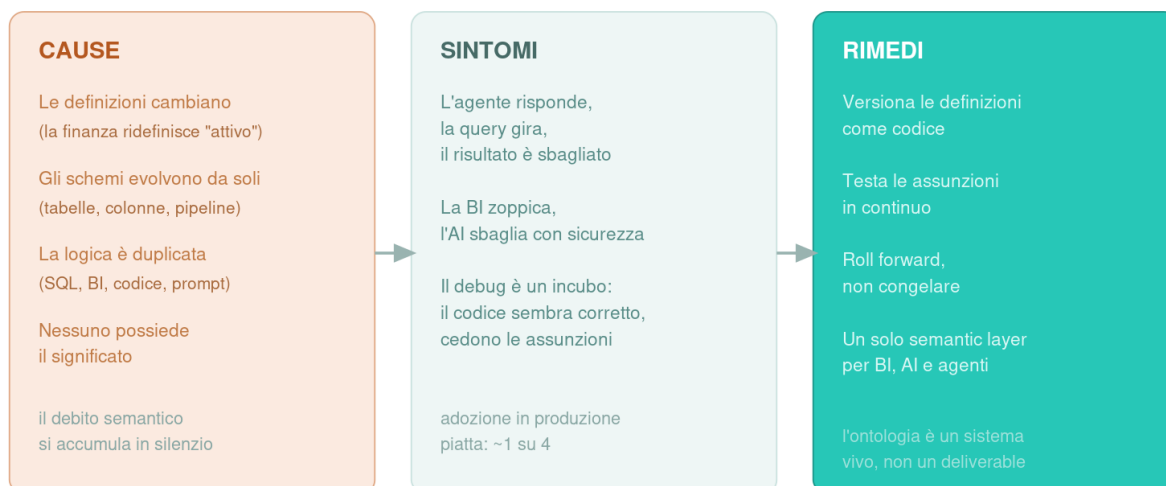


Fig. 11 – La deriva dell'ontologia.

E quando fallisce è difficile persino accorgersene, perché tutto sembra funzionare. L'agente risponde, la query gira senza errori, il risultato è formalmente valido, solo che non corrisponde più a ciò che il business intende. Il debug diventa un incubo, perché il codice sembra corretto, e in effetti lo è, a cedere sono le assunzioni sotto, che non valgono più. Un piccolo cambio in una definizione, "contratto attivo" che da oggi esclude i rinnovi anticipati, si propaga come un errore invisibile in ogni risposta costruita su quella parola, e nessuno lo vede finché qualcuno, per caso, non confronta un numero con la realtà e trova che non torna. È il motivo per cui i progetti che sembravano riusciti stallano dopo il go-live: il problema si è spostato dalla tecnologia al modello operativo, ed è lì che va spostato l'investimento, dal costruire una volta al mantenere sempre.

I team che non cadono nella trappola fanno poche mosse concrete e documentate. Trattano i cambi di significato come cambi di codice, versionati, revisionati, testati. Versionano esplicitamente le definizioni di business. Testano in continuo le assunzioni semantiche, con controlli automatici sui vincoli e sulle relazioni. Quando una definizione cambia non congelano, gestiscono la transizione con finestre di deprecazione e percorsi di migrazione. E usano un solo livello semantico condiviso tra BI, AI e agenti, così che le incoerenze emergano subito invece di annidarsi. L'ontologia va trattata come un sistema vivo, non come un deliverable di progetto che consegna e archivi. Evolve perché è connessa, non isolata.

La deriva è difficile da fermare per ragioni più organizzative che tecniche. Le ontologie vengono trattate come artefatti statici, con uno sforzo concentrato all'inizio e poi zero manutenzione, in pura logica di progetto. Il data stack e l'ontologia vivono in mondi separati, così il livello semantico non sa che i dati sono migrati su un nuovo warehouse. La competenza per metterci le mani è rara e costosa, e questo isola lo strato su un piccolo gruppo di specialisti che quasi nessuno consulta, col paradosso che meno gente lo guarda, meno si vede la deriva, meno pressione c'è a mantenerlo. E ogni aggiornamento rischia di rompere qualcosa a valle, una query che smette di girare, una dashboard con numeri diversi, così l'incentivo naturale diventa congelare tutto, e congelare è esattamente ciò che produce la divergenza.

Lo stesso divario ha un'altra faccia, la distanza tra chi costruisce i dati e chi li usa. I data warehouse li costruiscono gli ingegneri, le dashboard le costruiscono gli analisti, e in mezzo si apre una voragine, il livello semantico, quasi sempre fatto a mano, mal documentato, [perennemente in ritardo](#). Per costruire quello strato un ingegnere deve capire ogni chiave esterna, ogni catena di relazioni, ogni caso limite in cui una giunzione cambia natura, poi intervistare gli stakeholder per le domande di business, poi progettare le tabelle larghe che appiattiscono un modello relazionale complesso in qualcosa che uno strumento di analisi possa interrogare senza sei giunzioni. Giorni o settimane, per uno schema di media complessità, e un lavoro che il momento dopo comincia già a invecchiare. La direzione è generare questo strato in modo assistito, comprimendo settimane in ore, ma il punto resta identico a prescindere dallo strumento: quel ponte tra dati e significato è dove nascono i costi, e trattarlo come un lavoro una tantum è la garanzia che diventi debito.

La stessa intuizione sta dietro all'[Open Knowledge Format](#), lo standard aperto proposto da Google, sintetizzato con un'analogia efficace, un Git per la conoscenza. Come Git ha standardizzato versioning, portabilità e collaborazione sul codice, l'idea è fare lo stesso con la conoscenza organizzativa, oggi frammentata tra wiki, gestori di documenti, cataloghi dati, repository, e soprattutto nella testa delle persone più esperte. Il collo di bottiglia dell'AI si è spostato dall'intelligenza del modello al contesto, e il contesto è conoscenza organizzata, versionabile, portabile, che oggi quasi nessuno gestisce come gestisce il proprio codice.

12 Il governo del significato

Presa sul serio, la parola ontologia ti porta molto lontano dalla scelta di un prodotto o di un database. Ti porta in sala del consiglio, perché le decisioni vere riguardano chi possiede le definizioni, più che la tecnologia, e le definizioni vincolano tutto ciò che sta a valle. [Tre domande](#) meritano un posto nell'agenda del board, e ogni CEO e ogni CIO dovrebbero saperci rispondere.

La prima, chi possiede il chart of entities. Il master data management è il candidato più vicino, ma raramente lavora al livello delle definizioni e delle relazioni; il chief data officer di solito è la sede giusta, ma il suo mandato spesso non copre ancora questa governance della struttura. La seconda, trattare le definizioni delle entità core come decisioni di rilevanza per il board, ma solo quelle che toccano ricavo, rischio, esposizione regolatoria, fiducia del cliente, delegando il resto. La terza è la più scomoda: cosa assume la vostra strategia AI riguardo alla struttura. Quasi tutte le strategie AI assumono buoni dati, quasi nessuna assume dati strutturati, e quel divario è esattamente il punto in cui i programmi di intelligenza artificiale si arenano dopo il lancio.

Il board non deve mettere le mani su ogni definizione, e lo sforzo va calibrato. Meritano l'attenzione del consiglio solo le entità che toccano ricavo, rischio, esposizione regolatoria, fiducia del cliente. Il resto si delega. È una governance minima ma amata, concentrata dove le conseguenze sono reali e leggera dove non lo sono, l'unico modo perché la governance della struttura non diventi essa stessa un collo di bottiglia burocratico. La definizione di "cliente" che scegli decide chi entra nei numeri di retention, chi cade sotto un obbligo regolatorio, chi un sistema di AI tratterà come bersaglio di una decisione. Una volta scritta nel grafo, smette di essere implicita e locale e diventa esplicita, interrogabile, vincolante su tutto ciò che sta a valle. Una scelta con un proprietario, una storia e delle conseguenze, e trattarla come un dettaglio tecnico da delegare a chi configura il database è il modo più comune di scoprire troppo tardi di aver delegato una decisione strategica.

Esiste però un rischio speculare, e lo prendo perché è la parte più profonda e meno detta di tutta la storia, quello che si può chiamare [imperialismo semantico](#). Quando un'azienda dichiara che la propria ontologia rappresenta la realtà, il pericolo diventa epistemologico e politico prima che tecnico, assunzioni nascoste, bias istituzionale, punti di vista schiacciati dentro un'unica categoria, interoperabilità forzata. E non emerge quando la tecnologia fallisce, emerge quando ha successo, mentre gli impegni ontologici che ha codificato restano invisibili. Nessuna ontologia è la realtà. È il resoconto ufficiale di ciò che un'organizzazione sa di sé, un resoconto, non il territorio. E proprio questo è il suo valore per chi deve rispondere delle decisioni, perché una probabilità non è governabile, un modello ispezionabile sì. La governance ha bisogno di un oggetto, e l'ontologia è quell'oggetto.

Qui devo sciogliere un nodo, perché prima ho chiesto un'unica fonte di verità semantica e adesso dico che l'ontologia universale non esiste, e sembra una contraddizione che si dissolve appena si distinguono due piani. Dentro un'azienda serve un'unica definizione di "cliente" attorno a cui allineare marketing, finanza e supporto, e quella coerenza interna è governabile perché è il resoconto che l'impresa dà di sé, mentre tra aziende, tra settori, davanti alla realtà, la stessa definizione non ha alcun diritto di dirsi vera, e pretenderlo è l'imperialismo semantico. La fonte di verità che serve, insomma, è locale e dichiarata e non cosmica, e l'errore da evitare non sta nell'avere una definizione condivisa in casa, sta nello scambiarla per la struttura del mondo e nell'imporla a chi ha buone ragioni per vederlo diversamente.

Il problema esplode proprio adesso per una ragione che sta nello stesso crollo dei costi che ho salutato prima come buona notizia. Finché costruire un'ontologia richiedeva specialisti e mesi, i modelli semantici erano pochi e preziosi, e la scarsità li teneva sotto controllo. Adesso che generarne uno costa quasi nulla, [ogni dataset, ogni piattaforma, ogni agente, ogni reparto](#) può produrre la propria struttura, e il rischio diventa la proliferazione di tanti modelli internamente coerenti e reciprocamente disallineati, altro che la convergenza verso un linguaggio comune. La frammentazione del significato è più difficile da governare della sua assenza, perché un'azienda senza struttura sa di non averla, mentre un'azienda con dieci strutture che non si parlano è convinta di averla.

La risposta, allora, non è costruire l'ontologia unica e definitiva, che non esiste e non può esistere, è governare la pluralità. Rappresentazioni multiple, legittime, a volte incompatibili, rese visibili, tracciabili, navigabili. Una frase la tengo per chiudere il ragionamento: il futuro appartiene meno alle ontologie universali che alle architetture capaci di navigare la pluralità ontologica senza collassare nell'incoerenza. Non eliminare Babele, imparare ad abitarla. Sul piano operativo si traduce in tre livelli di maturità che le aziende sottovalutano sistematicamente, l'interoperabilità tecnica che fa parlare i sistemi, l'intelligenza documentale che riduce il carico cognitivo con LLM e recupero, e la governance collaborativa della conoscenza, i rituali umani con cui si negozia il significato, si risolvono i conflitti, si impara insieme. Le organizzazioni investono sui primi due livelli dando per scontato che il terzo emerga da solo. Non emerge da solo, e senza il terzo i primi due generano complessità senza resilienza.

Il nocciolo che regge tutto il resto è che [la semantica non è una proprietà del dato](#). Un LLM è probabilistico, predice sequenze di parole. Un'ontologia è deterministica, impone vincoli logici binari. Il significato è un'altra cosa ancora, emerge dall'interazione tra persone, strumenti e contesto, e non risiede né nei token né nei vincoli. Quando in una rete di attori succede un guasto non previsto, la via d'uscita non la trova l'algoritmo, la trova la squadra che comunica, mobilita conoscenze eterogenee e improvvisa. Il vero rischio è la fiducia mal riposta che disarmo la nostra capacità collettiva di dare senso, più che un'AI troppo potente. La macchina esegue correlazioni, l'umano si assume la responsabilità, e confondere i due piani è il modo più elegante di perdere il controllo senza accorgersene.

Un avvertimento in più meriterebbe una riflessione a parte, quello che chiamo il taylorismo cognitivo dell'AI. Ieri si meccanizzavano i corpi scomponendo il lavoro in gesti elementari, oggi si tenta di meccanizzare la conoscenza scomponendo la collaborazione in flussi automatizzabili, e l'umano nel ciclo diventa spesso cosmetico, un timbro su una decisione già presa dalla macchina. Inietta un'ontologia rigida dentro un sistema probabilistico a più attori e ottieni una risposta rivestita di sicurezza formale, avvolta nel vocabolario ufficiale, ma scollegata dalle pratiche reali distribuite tra le persone. Doppio rischio, la falsa sicurezza e l'invisibilizzazione dei conflitti semantici, perché i deboli segnali di un disaccordo interpretativo vengono levigati in una risposta uniforme. Anche il recupero aumentato cattura solo un'istantanea, e la documentazione è sempre parziale e datata. Le macchine processano segnali, [gli umani interpretano segni](#), e non confondere la mappa col territorio, in questo campo, vale come regola di sopravvivenza. Governare non vuol dire eliminare le differenze, vuol dire renderle visibili e navigabili.

Un ultimo strato riguarda chi tiene in mano la conoscenza. Un knowledge graph è la codifica della conoscenza proprietaria di un'impresa, la sua ossatura semantica, e questo lo rende un tema di sovranità prima ancora che di tecnologia. Un'azienda che affida il proprio significato a uno strato governato da altri, dentro una piattaforma che non controlla, sta prendendo una decisione strategica travestita da scelta tecnica. L'impresa cognitiva, quella che apprende e ragiona sui propri dati, ha bisogno di sapere dove vive la sua ontologia, chi la può leggere, chi la può cambiare. Per un'azienda italiana ed europea la cornice normativa rende tutto meno teorico: l'[AI Act](#) chiede sistemi di cui si possa spiegare il comportamento, e per i casi ad alto rischio quella spiegabilità diventa una condizione per operare, non un optional da concedere a fine progetto. Un modello ispezionabile è più difendibile di una probabilità, e ricostruire la catena di relazioni che ha portato a una decisione, il conto, il dispositivo, la controparte, permette di rispondere a un regolatore che chiede conto. Le norme più recenti sull'accesso ai dati vanno oltre, introducono il diritto di contestare una decisione automatizzata e di chiederne una revisione umana, il che rende la tracciabilità un obbligo prima ancora che un vantaggio. E i codici di corporate governance chiedono ai consigli di amministrazione di sapere spiegare come funzionano i sistemi su cui l'azienda si affida. La qualità rende una risposta corretta, la struttura la rende difendibile, e davanti a un regolatore la differenza tra le due è tutto.

13 Dal problema alla struttura

Resta la cosa più concreta di tutte, quella con cui si apre e si chiude ogni progetto vero: da dove si comincia. E la risposta che ho maturato [dai tavoli dove lavoro](#) è quasi il contrario di quello che ci si aspetterebbe da un articolo che ha appena dedicato quindicimila parole all'eleganza delle ontologie. Non si comincia dall'ontologia. Si comincia dal problema di business, quello che tiene sveglia la direzione la notte, e si risale da lì fino alla struttura che serve per scioglierlo. Un knowledge graph si giustifica per il problema che risolve, mai per l'eleganza del grafo, e contare le entità dentro il grafo non è un risultato, ridurre il tempo con cui si risponde a una domanda di business lo è.

Il primo lavoro, prima di qualsiasi grafo, è una diagnosi onesta della maturità del dato. Disponibilità, qualità, accessibilità, governance, e soprattutto frammentazione, quante sorgenti diverse, quanti sistemi che chiamano la stessa cosa con nomi diversi, quanta conoscenza vive solo nella testa delle persone. Se la maturità del dato è bassa e le sorgenti sono sparse su sette sistemi senza alcun consolidamento, quella è la cosa da indirizzare prima, non un caso d'uso predittivo da vetrina. Un knowledge graph, in questo quadro, è quasi sempre un abilitatore di fondo, il pezzo su cui si appoggiano gli altri, non il primo quick win da mostrare in un comitato per raccogliere applausi.

I numeri su questo punto sono impietosi e vanno detti chiari, perché smontano l'illusione che il problema sia il modello. Circa l'ottanta per cento del lavoro di un data scientist se ne va in pulizia dei dati, risoluzione delle entità, mappatura degli schemi, non in modellazione, la tassa nascosta che si paga quando la semantica è implicita e incoerente. E in un sondaggio del 2025 sui responsabili di programmi AI enterprise, due terzi indicano l'inconsistenza dei dati e la mancanza di allineamento semantico come la barriera numero uno al passaggio in produzione, sopra qualsiasi problema di modello o di infrastruttura. Il caso classico è l'identificativo del cliente chiamato `customer_id` in un sistema, `user_id` in un altro, `id` in un terzo, e nessuna potenza di calcolo che risolva ciò che è, in radice, una mancanza di coerenza semantica. Aggiungere modelli più grandi sopra questo strato non lo cura, lo amplifica, perché un agente prende quell'incoerenza e la propaga a velocità di macchina.

Aiuta distinguere i casi d'uso per natura, perché non si finanziano e non si comunicano allo stesso modo. Ci sono i quick win, alta prontezza, valore in poche settimane, quelli che costruiscono fiducia e slancio. Ci sono i casi strategici, alto impatto ma orizzonte di sei o dodici mesi. E ci sono i casi di fondo, gli abilitatori trasversali, i prerequisiti su cui poggiano tutti gli altri, e il knowledge graph vive quasi sempre qui. Il problema tipico è proprio questo, un abilitatore di fondo ha il ritmo lento e il valore differito di un'infrastruttura, mentre le aspettative attorno all'AI chiedono il ritmo veloce e il valore immediato di un quick win. Chi finanzia un knowledge graph aspettandosi applausi al primo trimestre lo abbandona al secondo, chi lo tratta per quello che è, una fondazione, raccoglie sui casi che quella fondazione abilita nei dodici o diciotto mesi successivi. Sapere in quale delle tre categorie stai mettendo un progetto è metà del suo destino.

Da qui il percorso ha una forma che le esperienze reali disegnano in modo abbastanza concorde. Si parte da un dominio circoscritto e ad alto valore, modellando l'essenza del business, un'identità cliente, una gerarchia di prodotto, un caso di rilevazione frodi, ed eliminando lì la polisemia, un termine con più significati, e i sinonimi, più termini per un significato. Si dimostra il ritorno su quel dominio prima di allargare. Poi si rompono i silos, portando l'ontologia a livello aziendale e mappandoci sopra i sistemi esistenti, così che le query attraversino le funzioni senza integrazioni fatte a mano ogni volta. Poi si itera sulla conoscenza, lasciando che algoritmi sul grafo facciano emergere relazioni implicite e segmenti nascosti. E infine, più una frontiera che una tappa raggiunta, il grafo diventa il motore di ragionamento a cui si collegano i modelli, che smettono di indovinare e interrogano una struttura deterministica. Questo arco richiede dai dodici ai diciotto mesi di investimento focalizzato per arrivare a maturità, e chiunque prometta di comprimerlo in un trimestre sta vendendo qualcos'altro.

Il percorso di maturità

Da un dominio circoscritto al motore di ragionamento. Dodici-diciotto mesi, non un trimestre.

↑ maturità - valore che si accumula sui casi abilitati

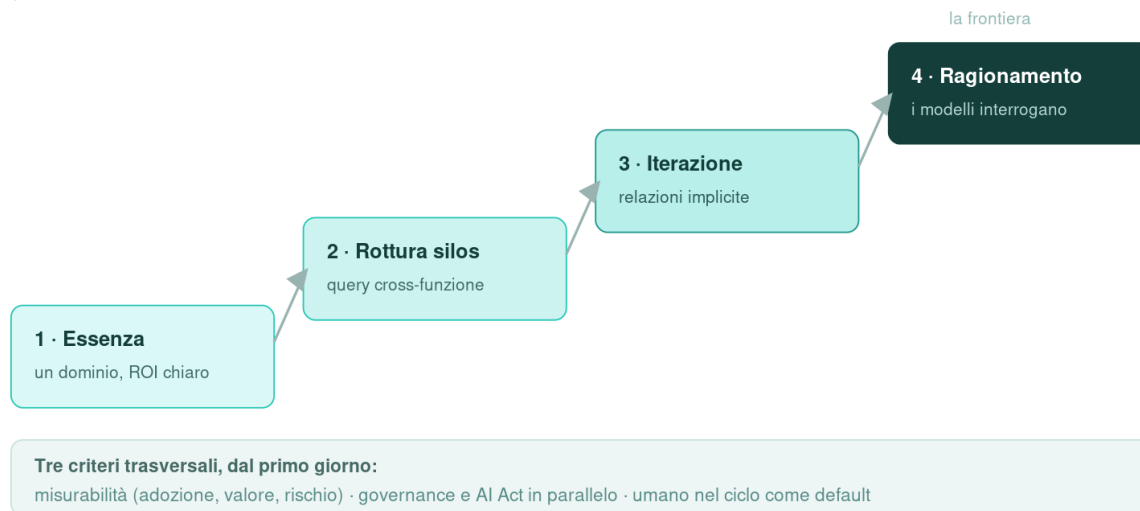


Fig. 12 – Il percorso di maturità.

Ogni tappa ha un esito che si può mostrare, e questo tiene vivo il finanziamento del programma. La prima produce un'applicazione sostenuta dall'ontologia con un ritorno chiaro su un dominio solo, e un modello ripetibile per allargarsi. La seconda produce query che attraversano le funzioni senza costruire un'integrazione su misura ogni volta. La terza fa emergere relazioni implicite e previsioni come nuovi spunti, cose che nessuno aveva chiesto perché nessuno sapeva di poterle chiedere. La quarta, la frontiera, chiude il cerchio, l'insight validato dall'AI rientra nell'ontologia come nuova regola e il sistema impara su se stesso. I ritorni raccontati da chi è arrivato ai primi stadi non sono simbolici, un operatore industriale che cresce il core business di oltre dieci punti, una rete energetica che fa analisi di causa in secondi invece che in giorni, un servizio che passa da giorni di indagine a venti secondi per risalire alla radice di un problema. Non serve avere l'ontologia completa domani, serve partire da un dominio circoscritto e ad alto impatto, dimostrare il valore, allargare da lì. Gran parte delle organizzazioni sta ai primi due stadi, e va benissimo, purché sappia dove sta e dove vuole arrivare.

Sopra questo percorso corrono tre criteri, diversi dalle fasi perché valgono in ognuna, e vanno tenuti dal primo giorno. Il primo, la misurabilità: nessun caso d'uso parte senza una baseline misurata e un obiettivo esplicito, perché se non sai dov'eri prima non puoi dichiarare di essere migliorato, e le metriche vanno prese su tre famiglie, adozione, valore e rischio, mai solo l'adozione, perché "l'AI usata cinquemila volte" non è un risultato. Il secondo, la governance: comitato, ruoli, policy, e per i casi ad alto rischio la verifica di conformità all'AI Act vanno impostati in parallelo al primo caso d'uso, non il giorno dopo, perché la governance rincorsa costa il doppio. Il terzo, l'umano nel ciclo come default: controllo umano esplicito almeno nel rollout, con validazione prima dell'invio, punteggi di confidenza, escalation automatica, soprattutto dove ci sono effetti su clienti, dipendenti o conseguenze legali.

Alcune trappole ricorrenti ho visto affossare progetti buoni, e riconoscerle in tempo vale più di qualsiasi metodologia. La roadmap scritta prima della diagnosi, che produce liste di casi d'uso scollegate dalla maturità reale. Il proof of concept che non muore mai, tenuto in vita per mesi senza un go o un no esplicito, quando andrebbe chiuso con una decisione binaria entro poche settimane. Le metriche di

vanità, contare quante volte è stato usato uno strumento invece del tempo risparmiato o degli errori evitati, che qui significa contare le entità del grafo invece di misurare quanto più in fretta si risponde a una domanda che conta. E il tecnicismo che esclude, gestire il programma con un linguaggio da addetti ai lavori, embedding, reasoner, retrieval, che taglia fuori proprio i responsabili di business che dovrebbero guidarlo. Un articolo su ontologie e knowledge graph corre lo stesso rischio, e per questo ho provato a tenerlo leggibile da chi lo deve guidare, non solo da chi lo implementa.

Per capire se la fatica ti riguarda adesso o puoi rimandarla, cinque segnali distillati dalle esperienze enterprise funzionano da autodiagnosi, la frammentazione dei dati su più sistemi, i requisiti di compliance alti, le catene decisionali lunghe, l'eterogeneità forte dei dati, il costo alto di una risposta sbagliata, e se ne riconosci tre o più, l'ontologia diventa una necessità competitiva più che un lusso da laboratorio, e l'alternativa, continuare a costruire ponti su misura tra un silo e l'altro, uno alla volta, costosi e fragili, è la strada che sembra più economica oggi e si rivela la più cara domani. La consistenza semantica, a quel punto, smette di essere un progetto e diventa una funzione della piattaforma.

Tre tipi di organizzazione adottano l'ontologia a scala prima delle altre, e conviene sapere se ci si riconosce. Le piattaforme con moltissimi utenti, dove ogni nuova funzionalità, senza un'ontologia unificata, costringe a reintegrare il grafo da zero e le reti di frode sfruttano le crepe tra un pezzo e l'altro. Le industrie fortemente regolate, banche, sanità, assicurazioni, che devono spiegare le proprie decisioni tracciando le catene di relazione tra conti, dispositivi, controparti. E le reti operative complesse, supply chain, logistica, reti energetiche, telecomunicazioni, dove l'analisi di causa passa da giorni a secondi quando il valore vive nelle relazioni e non nei singoli record. Il filo che le unisce è lo stesso, il valore sta nelle relazioni, non nei dati isolati, e più il tuo business assomiglia a una rete, prima l'ontologia smette di essere un lusso.

14 Il grafo latente

Torno alla parola "cliente" con cui ho aperto, perché adesso quella scena si legge diversamente. Le tre definizioni che convivono al tavolo tra vendite, finanza e supporto sono l'ontologia della tua azienda così com'è ora, implicita, incoerente, mai scritta, e proprio per questo impossibile da governare. Il grafo latente non lo devi creare dal nulla, esiste già, distribuito nelle teste delle persone e nelle giunture tra i sistemi, e ogni giorno produce decisioni al posto tuo.

Per anni te lo potevi permettere, perché a leggere quei dati c'erano persone che rammendavano l'ambiguità con il buon senso, sapevano di quale "cliente" si stesse parlando e aggiustavano a mente. Il buon senso, dentro un modello, non c'è. Un agente eredita le tue definizioni così come sono, coerenti o contraddittorie, scritte o lasciate al caso, e le applica alla velocità e alla scala di una macchina, moltiplicando l'ordine che hai messo o il disordine che hai lasciato.

Tanto vale scegliere adesso quelle definizioni, mentre a rileggerle ci sono ancora persone capaci di correggerle. Il momento in cui gli agenti cominceranno a ragionare sul serio sui tuoi dati non è lontano, e quel giorno il grafo latente diventerà il tuo grafo reale, per come lo avrai coltivato o per come lo avrai lasciato crescere da solo.

Fabio Lalli

Imprenditore, advisor e speaker su AI e innovazione. ZeroFive.AI