UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Dipartimento di Informatica

Corso di Laurea in Informatica

Tesi di Laurea

in

Basi di Dati

Annotazione Semantica di pagine Web di eCommerce mediante Schema.org ed Applicazione ad un Caso Aziendale

Relatrice:
Prof.ssa Claudia d'Amato

Laureando:
Francesco Maria Reggio

Correlatori:
Dott. Roberto Barile
Dott.ssa Eufemia Lella

Anno Accademico 2023 - 2024

Sommario

La disponibilità e la fruibilità dell'informazione sono due caratteristiche fondamentali che un servizio di ricerca sul Web deve garantire per poter rispondere in modo efficace alle richieste dell'utente. Con l'aumento esponenziale del volume dei dati disponibili sul Web, questo compito è diventato sempre più importante e complesso.

Il successo stesso di un motore di ricerca dipende intrinsecamente dalla propria capacità di fornire informazioni rilevanti all'utente, in modo tale che esse siano rispondenti quanto più possibile alla richiesta dell'utente, vengano proposte tra i primi risultati e vengano presentate in maniera facilmente fruibile. Per questo motivo, sono state sviluppate delle soluzioni tecnologiche che puntano a rendere interpretabile anche dalle macchine l'informazione contenuta all'interno delle pagine Web. In questo modo, si facilita l'estrazione e la presentazione delle informazioni rendendole più pertinenti e utili per l'utente, migliorando così l'efficacia della navigazione e della ricerca online.

Tra le tecnologie implementate è di particolare rilevanza l'*annotazione semantica*, che arricchisce il contenuto delle pagine Web con informazioni che descrivono il significato dei dati in un formato adatto all'elaborazione automatica. Con l'annotazione semantica si esplicitano le informazioni rilevanti attraverso i diversi formati di marcatura disponibili e facendo riferimento a vocabolari condivisi. Le informazioni diventano così rielaborabili più facilmente dai motori di ricerca, che possono quindi generare risultati più mirati per le richieste dell'utente e presentarli in modo più efficace e informativo.

L'obiettivo di questo lavoro di tesi è valutare l'impatto che l'adozione dell'annotazione semantica ha sulle pagine Web. In particolare, si analizza come il *Web*

semantico e l'uso di vocabolari (organizzati come dati strutturati) condivisi, come ad esempio *Schema.org*, possano creare vantaggi sia per l'utente che nell'elaborazione delle informazioni. Da un lato, l'annotazione semantica consente di migliorare il posizionamento delle pagine più rilevanti nei risultati di ricerca e la presentazione delle informazioni al loro interno, dall'altro lato favorisce una comprensione più profonda dei contenuti da parte dei motori di ricerca e dei sistemi automatici, consentendo richieste più specifiche e una migliore elaborazione dei dati.

Lo studio è condotto in collaborazione con l'azienda Sidea Group S.r.l., il cui ruolo è stato di importante rilevanza. L'azienda ha fornito l'ambiente di lavoro, in cui è stato possibile interfacciarsi con le tecnologie e le soluzioni software necessarie, nonché il supporto e le competenze del gruppo di lavoro aziendale.

Con l'obiettivo di dimostrare quantitativamente che gli strumenti di ricerca semantica siano in grado di rispondere in modo efficace a query molto specifiche e complesse, sfruttando i dati strutturati, è stato progettato un framework applicabile a diversi contesti di utilizzo. Il framework è diviso in fasi di *annotazione*, *raffinamento* e *valutazione*, che prevedono la costruzione di annotazioni semantiche, il loro successivo raffinamento e quindi la valutazione delle prestazioni derivanti dall'utilizzo di tali annotazioni con sistemi di ricerca semantici.

Il framework viene quindi applicato ad un sito reale di eCommerce, reso disponibile dall'azienda Sidea Group S.r.l. in modo tale da mostrare come i dati strutturati associati ai prodotti possono essere impiegati direttamente per la ricerca di prodotti. Migliorando la correttezza dei risultati presentati all'utente, si può potenzialmente incrementare il numero di vendite e di visite al sito. È pertanto di fondamentale importanza per un sito di eCommerce assicurarsi che vengano restituiti al cliente i risultati più attenenti alla propria interrogazione.

A valle dello studio, i risultati hanno evidenziato che i sistemi di ricerca semantica hanno la capacità di rispondere in modo specifico alle interrogazioni, utilizzando direttamente i dati strutturati, mentre i sistemi di ricerca basati su parole chiave, sebbene più usabili, sono generalmente inefficienti a elaborare query complesse e specifiche. Il sistema basato sulla traduzione di query testuali con LLM combina efficacemente i vantaggi di entrambi gli approcci, con ottime prestazioni.

Indice

Capitolo 1

Introduzione

La quantità di informazioni presente sul Web è vasta e in continua evoluzione, con più di 5 miliardi di utenti di Internet che creano, modificano ed eliminano pagine nel Web ogni giorno [18]. È difficile immaginare quante di queste possano contenere informazioni utili ed attinenti a ciò che viene effettivamente cercato. È proprio questo il ruolo che svolgono i motori di ricerca. A partire dalla vasta, indiscriminata mole di informazioni disponibile sul Web, riescono ad organizzare le pagine e ad elaborare i dati interni ad esse per trovare quelle che soddisfano maggiormente le interrogazioni degli utenti.

Nel 2016, Google ha stimato che nel Web erano presenti ben 130,000 miliardi di pagine [30]. Soltanto una parte di queste pagine, nell'ordine delle decine di miliardi, viene effettivamente indicizzata dai motori di ricerca e può essere esposta tra i risultati di una corrispondente ricerca da parte dell'utente [31]. I motori di ricerca danno sempre più importanza alla qualità delle pagine Web che vengono indicizzate rispetto al loro numero. È per questo motivo che, nonostante il numero totale di pagine presenti nel Web aumenti di anno in anno [7], il numero di pagine Web effettivamente indicizzate dai motori di ricerca può sia aumentare che diminuire, con gran parte di questa variabilità attribuibile a cambiamenti principali nell'architettura e negli algoritmi di indicizzazione [31].

Infatti, i motori di ricerca cercano di migliorarsi nel tempo con l'obiettivo di

proporre all'utente le pagine più pertinenti e allo stesso tempo riducendo la visibilità delle pagine non pertinenti o spam, come veniva svolto già dall'algoritmo di PageRank, introdotto alla nascita di Google [22] e ancora utilizzato in una forma più evoluta [9]. Da allora, i motori di ricerca sono diventati sempre più sofisticati nel rilevare e valutare i contenuti delle pagine Web e un aspetto significativo di questa evoluzione è stata l'introduzione del *Web semantico* [12], che si pone l'obiettivo di rendere le informazioni nel Web *machine-readable* e *machine-understandable*, ovvero strutturate in un formato che le macchine possono facilmente interpretare ed elaborare cogliendone in qualche modo il significato [4].

I *metadati* svolgono un ruolo centrale nel Web attuale, in quanto sono annotazioni che descrivono al meglio l'informazione. Tramite i metadati, si organizzano e si strutturano i dati nel Web. La loro potenzialità è stata riconosciuta già con l'introduzione di HTML 2.0 nel 1995, che ha fornito i primi strumenti per incorporare metadati attraverso l'elemento `meta` e attributi come `rel` e `rev` [3]. Questi strumenti erano primitivi e alquanto limitati rispetto alle tecnologie attuali, in quanto erano principalmente destinati alla presentazione e formattazione dei contenuti senza fornire una descrizione semantica delle informazioni [13]. Nonostante ciò, hanno gettato le basi per un'evoluzione continua nella gestione e nella interpretazione dei dati.

Allo stato attuale, i metadati fanno riferimento a vocabolari condivisi che permettono la standardizzazione ma soprattutto la condivisione della descrizione semantica delle informazioni. Nei contesti più complessi, si utilizzano ontologie che forniscono una descrizione logica formale, consentendo la descrizione di concetti complessi e la rappresentazione di molteplici relazioni con diverso significato tra concetti. Mentre i vocabolari descrivono termini e le loro proprietà, le ontologie definiscono in modo formale concetti e proprietà.

Sono state sviluppati nel tempo diversi vocabolari standardizzati di metadati che hanno facilitato la modellizzazione dei dati strutturati per il Web in molteplici settori come negozi online, editoria, social media e così via [16]. Nel corso degli anni ci sono state diverse iniziative per aggiungere elementi semantici all'interno delle pagine Web, come OHTML, che però non ha avuto una larga diffusione a causa della

complessità di integrazione dei dati strutturati ai commenti HTML [13]. Nonostante ciò, le basi erano già fissate con l'identificazione di una rete multi-relazionale composta di oggetti identificati univocamente. Successivi sforzi coordinati a livello globale per unificare diversi approcci complementari di dati strutturati hanno incluso lavori su *Microformats*, *RDFa* e *Microdata* [13]. Uno di questi progetti, iniziato nel 2011, è *Schema.org* [16].

Schema.org è nato da una collaborazione tra Google, Microsoft, Yahoo e Yandex — organizzazioni che, sebbene siano tradizionalmente concorrenti commerciali, hanno deciso di collaborare su un unico modello di dati strutturati per il Web al fine di migliorare i loro prodotti di ricerca, con il supporto di altri contributori come gruppi scientifici e membri del W3C[1]. Questo sforzo congiunto ha portato alla creazione di uno schema di markup dei dati unificato che ha facilitato l'integrazione di dati strutturati nelle pagine Web, superando le difficoltà legate alla formattazione diversificata del markup causate da una adozione non omogenea di diversi vocabolari e dallo sviluppo di numerosi standard non interoperabili [12].

Oggi, milioni di siti Web e applicazioni fanno uso di Schema.org per assistere nella ricerca e nel recupero delle informazioni, a partire da assistenti virtuali come Alexa[2], Google Assistant[3], Siri[4] e Cortana[5], che utilizzano le annotazioni delle pagine Web per rispondere alle interrogazioni degli utenti [16], per passare ai portali di film come IMDb[6] e Rotten Tomatoes[7], ai siti di eCommerce come eBay e Alibaba, ai siti di notizie come The New York Times e BBC, fino alle piattaforme di video come YouTube e Dailymotion [12]. In tutti questi casi, le annotazioni delle risorse sono essenziali per migliorare la qualità ma soprattutto il ritrovamento dei dati e

[1] Il World Wide Web Consortium (W3C) è un'organizzazione internazionale no-profit che sviluppa standard per il Web. Fondato dall'inventore del Web Tim Berners-Lee, il W3C lavora con membri, staff e pubblico per ottimizzare il potenziale del Web. `https://www.w3.org/about/`.

[2] `https://www.alexa.com/`

[3] `https://assistant.google.com/`

[4] `https://www.apple.com/it/siri/`

[5] `https://www.microsoft.com/en-us/cortana`

[6] Internet Movie Database (IMDb), sito specializzato su recensioni di contenuti multimediali: `https://www.imdb.com/`

[7] Anche Rotten Tomatoes è principalmente un sito di recensioni su film e serie TV: `https://www.rottentomatoes.com/`

dei servizi erogati.

Schema.org ha offerto una soluzione integrata che ha contribuito a semplificare e uniformare il vocabolario impiegato nell'annotazione semantica, migliorando la qualità, l'interoperabilità e la consistenza dei dati strutturati nel Web [12].

I dati strutturati hanno un ruolo centrale nella presente tesi perché sono utilizzati per migliorare l'accuratezza e la pertinenza delle interrogazioni sui prodotti di eCommerce tramite l'annotazione semantica delle pagine Web. Si valuta quindi l'impatto dell'utilizzo diretto dei dati strutturati nei sistemi di ricerca, rispetto a sistemi che non utilizzano annotazioni semantiche.

1.1 Contesto

La forte competizione nella ricerca sul Web ha spinto le aziende ad andare oltre il semplice *ranking*[8] dei risultati per migliorarli. Una tecnica utilizzata inizialmente da Yahoo e successivamente da Google è stata quella di arricchire i risultati di ricerca con dati strutturati provenienti dalla pagina stessa, ottenendo così i *rich snippets*, risultati arricchiti con informazioni aggiuntive [12]. Il Web tradizionale si sta gradualmente evolvendo verso un Web semantico, in cui le informazioni interne alle pagine sono elaborabili anche dalle macchine per migliorare l'esperienza dell'utente [4].

Nel settore dell'eCommerce, l'adozione di tecnologie di annotazione semantica rappresenta un passo cruciale per migliorare la comprensibilità del contenuto delle pagine Web. Questa adozione ha l'obiettivo di facilitare un ritrovamento più efficace delle informazioni e delle pagine di interesse. Con i dati strutturati, i motori di ricerca e le piattaforme di eCommerce sono in grado di interpretare e organizzare meglio le informazioni sui prodotti, ottimizzando così la visibilità delle pagine e migliorando la correttezza e la completezza dei risultati di ricerca.

Un esempio pratico è la piattaforma di shopping online eBay, che ha intrapreso

[8] Posizionamento di una pagina Web nei risultati di ricerca.

una significativa trasformazione negli ultimi anni. Nel 2016, infatti, eBay ha annunciato importanti aggiornamenti nell'approccio ai dati strutturati, con l'introduzione di un sistema standard di identificazione univoca e di annotazione dei prodotti, in modo tale da migliorare le informazioni associate ad essi [6]. Man mano, l'approccio si è focalizzato sempre di più sul fornire informazioni dettagliate sui prodotti. Nel 2019, eBay ha ulteriormente affinato la sua strategia, concentrandosi sulla raccolta di dettagli dai venditori attraverso campi di specifiche estesi e incentivando la somministrazione di informazioni dettagliate per migliorare la visibilità delle inserzioni [28].

La Figura 1.1 schematizza il ruolo che svolgono le annotazioni organizzate come dati strutturati su eBay [6]. I clienti utilizzano il sito di eBay o piattaforme esterne per cercare i prodotti secondo le loro necessità. Sul sito di eBay, i dati strutturati aiutano i clienti a trovare gli articoli in modi diversi, a partire dalle pagine dei prodotti, dai filtri, dalle pagine di categorie, dalle offerte di eBay fino ad arrivare alla ricerca interna al sito e alle recensioni dei prodotti. Quindi, i dati strutturati permettono di rispondere alle interrogazioni degli acquirenti con l'articolo più rilevante grazie ad una descrizione del prodotto più ricca, a dettagli sui prodotti organizzati in un modo più consistente e alle recensioni, che sono ora annotate semanticamente. Da tutto ciò ne deriva una visibilità più alta e un traffico più elevato.

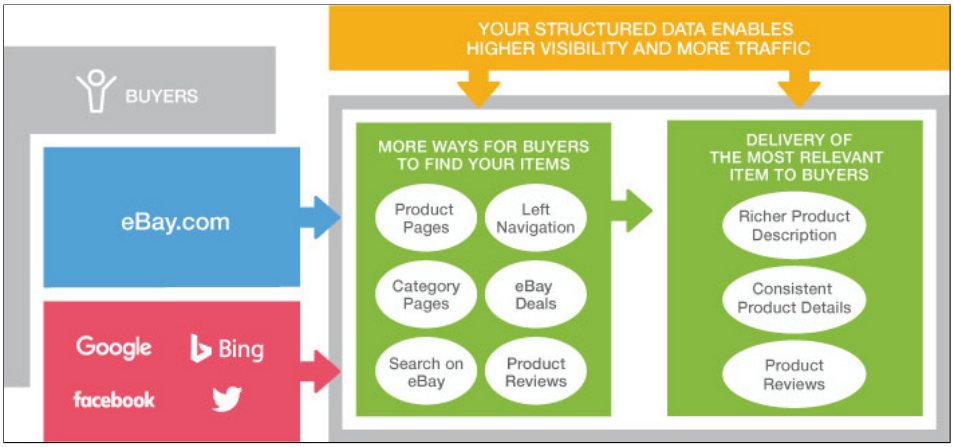

Figura 1.1: Grafico che mostra i benefici dei dati strutturati su eBay [6].

Con la presente tesi si è voluto studiare l'impatto e la modalità di applicazione dell'annotazione semantica focalizzandosi su uno scenario di utilizzo reale. Il caso

di studio analizzato si inserisce in un contesto aziendale, in cui la necessità di fornire al cliente un servizio che aderisca nel miglior modo possibile alle proprie esigenze rende il ruolo dell'annotazione semantica fondamentale.

Le aziende dedicano sempre più importanza alla *Search Engine Optimization* (SEO), un insieme di attività e processi finalizzati a migliorare l'indicizzazione di un contenuto sul Web e ad elevarne il posizionamento nei risultati dei motori di ricerca. Oltre al ranking, altrettanto utile è la percezione del *brand* e la qualità del sito. Google consiglia questo approccio per la SEO, ovvero di basare le decisioni di ottimizzazione, prima di tutto, su ciò che è meglio per i visitatori del proprio sito. Generare risultati di buona qualità, utilizzando i dati strutturati interni alla pagina garantisce che essi rispecchino maggiormente le necessità peculiari dei clienti [2].

È in questo contesto che si colloca la collaborazione con l'azienda Sidea Group S.r.l., società di consulenza informatica con sede a Bari che sviluppa per aziende terze soluzioni software, con particolare attenzione per la soddisfazione dell'utente. La divisione SIRIO (Sidea Research & Innovation), specializzata in ricerca e sviluppo, ha avuto un ruolo di primaria importanza nel fornire il supporto tecnico necessario per il lavoro svolto nella presente tesi, con l'aiuto di altre divisioni.

In particolare, l'azienda Sidea Group S.r.l. ha messo a disposizione un sito Web di eCommerce su cui applicare il framework progettato nel Capitolo 3, permettendo di avere il controllo completo del progetto in un ambiente locale su cui poter svolgere le operazioni necessarie per l'annotazione semantica, il raffinamento delle annotazioni e la valutazione delle prestazioni di ricerca.

A partire dal catalogo di un sito di eCommerce a disposizione dell'azienda, sono stati analizzati i dati strutturati dei prodotti e sono stati espansi annotando le informazioni mancanti. È stata quindi eseguita la fase di valutazione del framework, che ha permesso di poter confrontare due diversi insiemi di risultati a partire da query svolte su un catalogo di prodotti di un sito Web di eCommerce. Un insieme è stato generato da una ricerca basata su parole chiave, usando il motore interno al sito, e l'altro è proveniente da equivalenti query in linguaggio SPARQL, che fa uso diretto dei dati strutturati.

1.2 Motivazioni

L'idea centrale della tesi è confrontare le tecnologie di annotazione semantica con un caso reale di prodotto aziendale nel campo dell'eCommerce, ponendo particolare enfasi su come i vantaggi derivanti da tali tecnologie siano accessibili a tutti i tipi di aziende, non solo alle grandi multinazionali. Le tecniche di annotazione semantica, attraverso l'inserimento di dati strutturati nelle pagine Web, permettono infatti di migliorare notevolmente la comprensione e l'indicizzazione dei contenuti da parte dei motori di ricerca, con effetti positivi diretti sul posizionamento e sul numero di visitatori della pagina [11]. Questo significa che non solo i grandi attori del mercato, ma anche le piccole e medie imprese possono trarre vantaggio dall'adozione di queste tecnologie per competere efficacemente sul Web.

L'obiettivo del progetto è dimostrare che i benefici derivanti dall'utilizzo dell'annotazione semantica, se implementati correttamente, sono applicabili a qualsiasi organizzazione, indipendentemente dalle dimensioni o dalla complessità. Nel contesto dell'eCommerce, dato che l'annotazione semantica facilita la comprensione del contenuto delle pagine Web dai motori di ricerca, possono essere utilizzati formati strutturati anche per le interrogazioni, ad esempio tramite il linguaggio SPARQL che fa riferimento in modo diretto alle informazioni strutturate, a differenza della ricerca semplice basata su parole chiave. Dall'utilizzo dei metodi basati sull'annotazione semantica ne deriva una migliore corrispondenza tra le richieste dei potenziali clienti e i prodotti offerti dall'azienda, il che può tradursi in un incremento sostanziale delle visite e, proporzionalmente, di vendite [8].

1.3 Organizzazione della tesi

La presente tesi è strutturata in sei capitoli, ciascuno dei quali focalizzato su un argomento specifico. Si parte con il Capitolo 1, che rappresenta l'introduzione al lavoro di tesi. Si prosegue con il Capitolo 2, che formalizza lo stato dell'arte, fornendo le definizioni teoriche di base per la tesi. Si parte con la definizione di Web semantico, per poi evolvere verso concetti fondamentali come *RDF*, *RDF Schema*, *SPARQL* e *OWL*. Il capitolo approfondisce anche l'annotazione semantica e i principali formati di marcatura come *RDFa*, *Microdata* e *JSON-LD*.

Nel Capitolo 3 si descrive il framework progettato per l'annotazione semantica di risorse Web e per la valutazione dell'impatto sul ritrovamento di tali risorse da parte dei motori di ricerca. Si illustrano in dettaglio le tre fasi di *annotazione*, *raffinamento* e *valutazione* del framework e come si arriva gradualmente al calcolo delle prestazioni derivanti dall'utilizzo dei dati strutturati nella ricerca.

Il Capitolo 4 è dedicato all'applicazione al caso di studio in esame. In questo capitolo si applica il framework ad un sito Web di eCommerce messo a disposizione dall'azienda Sidea Group S.r.l., fornendo una descrizione degli strumenti sviluppati e utilizzati per arrivare alla misura quantitativa delle prestazioni dei due sistemi. Si parte con la parte di annotazione delle risorse nel sito, per poi proseguire con la costruzione del *corpus dei dati* inerente ai prodotti presenti nel sito oggetto dello studio. Quindi, viene discussa l'attuazione delle scelte di progetto che hanno portato alla creazione del corpus di query con i risultati esatti correlati, alla conversione in query in linguaggio naturale e in SPARQL, e le relative esecuzioni sul catalogo di prodotti. Viene inoltre proposto un approccio alternativo di esecuzione di query testuali traducendole in SPARQL con l'ausilio di *Large Language Model* (LLM) per l'interpretazione del linguaggio naturale.

I risultati finali derivanti dal caso di studio sono presentati e discussi nel Capitolo 5. In questo capitolo si valutano le performance delle query testuali e in SPARQL, misurate in termini di *precisione*, *richiamo* e *F1 measure*. Si confrontano i risultati quantitativi per determinare il metodo più efficace. Inoltre, si discutono i risultati derivanti dalla traduzione di query testuali in SPARQL e la loro successiva esecuzione sullo stesso catalogo di prodotti.

Infine, nel Capitolo 6 si raccolgono le conclusioni derivanti dallo studio. Viene fornita una sintesi dei risultati, si discutono i limiti dello studio e si suggeriscono possibili sviluppi futuri della ricerca.

Capitolo 2

Stato dell'Arte

Il presente capitolo fornisce le nozioni di base su cui si fonda il lavoro di tesi. Si inizia con l'introduzione del *Web semantico*, concepito come evoluzione del Web tradizionale per permettere una comprensione più profonda e contestuale dei dati da parte delle macchine. Successivamente, vengono illustrati i principali linguaggi di rappresentazione adottati come standard nel Web semantico, come RDF, RDFS, SPARQL e OWL. Il capitolo poi prosegue esplorando l'annotazione semantica, il ruolo dei vocabolari condivisi, come Schema.org, e i principali formati di annotazione, come RDFa, Microdata e JSON-LD.

2.1 Web semantico

Il Web è stato introdotto da Tim Berners-Lee nel 1991 come un sistema per accedere e condividere informazioni attraverso l'uso di hyperlink. Con il passare degli anni, il Web ha subito una significativa evoluzione, passando dalla prima generazione di contenuti statici del Web 1.0, dove gli utenti erano principalmente consumatori passivi di informazioni, all'era del Web 2.0, caratterizzata da un maggiore coinvolgimento degli utenti e dalla creazione di contenuti generati dagli utenti stessi.

Il *Web semantico* è stato ideato da Tim Berners-Lee nel 2001 per indicare una nuova forma del Web in cui l'informazione è annotata semanticamente in modo

tale che le macchine possano comprenderla e processarla. A differenza del Web tradizionale, progettato principalmente per essere leggibile dagli umani, il Web semantico è stato concepito con l'idea che i computer possano svolgere attività complesse tramite una comprensione attiva della semantica, e quindi del significato dietro i dati, i quali diventano interconnessi e maggiormente integrati all'interno del Web [4].

Due tecnologie importanti per lo sviluppo del Web semantico, già presenti alla sua concettualizzazione, sono *eXtensible Markup Language* (XML) e *Resource Description Framework* (RDF), rispettivamente per etichettare sezioni di testo e per associare un significato semantico a tali strutture [4].

In particolare, la Figura 2.1, elaborata da Berners-Lee nel 2002[1], mostra le tecnologie standard che compongono il Web semantico, sotto forma di una "torta a strati". Al livello più basso, si trovano le tecnologie di base per l'identificazione e la codifica dei contenuti, che includono gli *Uniform Resource Identifiers* (URI) per la denominazione, *Unicode* per la codifica dei caratteri, XML per la strutturazione dei documenti e i *namespaces* per garantire l'univocità dei nomi degli elementi.

I tre strati centrali forniscono una rappresentazione in termini di concetti, proprietà e individui, utilizzando RDF per modellare le informazioni, *RDF Schema* (RDFS) per definire classi e proprietà delle risorse RDF, e il *Web Ontology Language* (OWL) per la creazione di ontologie.

Gli strati superiori estendono l'espressività attraverso il *Rule Interchange Format* (RIF)[2] e il *Semantic Web Rule Language* (SWRL)[3], utilizzati per rappresentare regole inferenziali. A questi strati si aggiungono il Logic Framework, una logica unificatrice, e meccanismi per la Proof e la Trust, che sono essenziali per verificare la veridicità delle conoscenze dedotte e per convalidare l'origine dei dati utilizzando tecniche crittografiche. Tuttavia, va notato che gli ultimi strati non sono ancora completamente supportati da standard consolidati.

[1] `https://www.w3.org/2002/Talks/04-sweb/slide12-0.html`

[2] `https://www.w3.org/TR/rif-overview/`

[3] `https://www.w3.org/submissions/2004/SUBM-SWRL-20040521/`

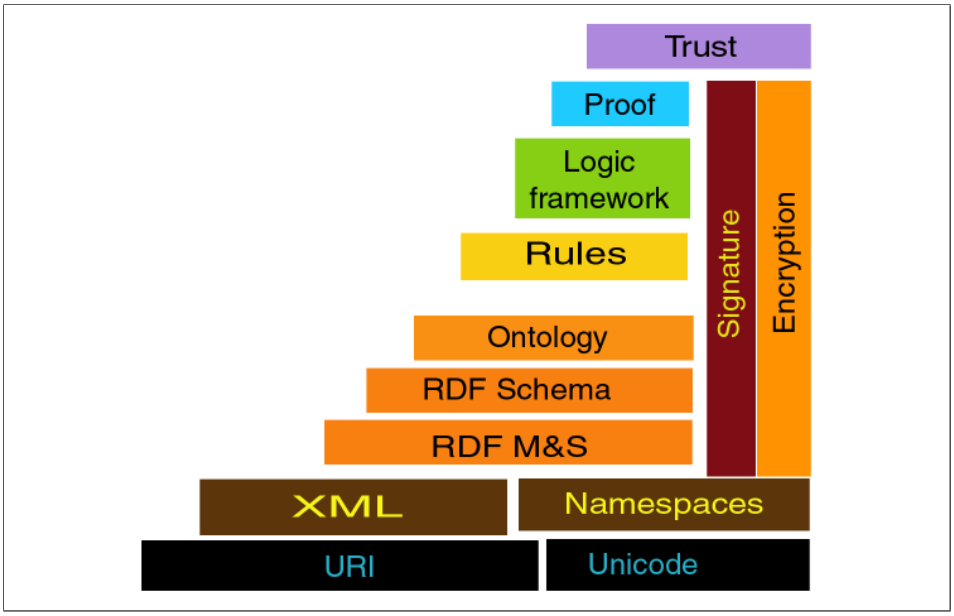

Figura 2.1: Schema che mostra la struttura a livelli con le componenti del Web semantico.

Nel linguaggio XML gli elementi sono annotati mediante tag in modo gerarchico. Vengono definite un insieme di regole per codificare documenti in un formato leggibile sia dagli umani che dalle macchine. XML consente a chiunque di creare i propri tag, utilizzando campi del tipo `<tag>` e `<label>` per annotare una sezione di pagina. I programmi possono quindi leggere queste informazioni ed elaborarle, a patto che riescano a interpretarvi il significato. Il linguaggio XML, infatti, viene usato unicamente per etichettare e strutturare i dati nei documenti, ma non fornisce informazioni sul significato di tali strutture, il cui ruolo è delegato al linguaggio RDF [4].

Nel 2006, l'idea del Web semantico è stata rivisitata nel *Web of Data*, ovvero Web dei dati, in cui si pone una maggiore enfasi sull'interconnessione globale dei dati, sotto forma di ontologie e dati, pur mantenendo la visione del Web semantico [27]. Con il termine *Linked Data*, o dati collegati, si intende un modo per creare dati interconnessi ed interpretabili dalle macchine, basati sugli standard del Web semantico precedentemente citati. Questi standard permettono di collegare dati provenienti da diverse fonti, mantenendo una struttura che facilita l'accesso, la condivisione e il riutilizzo delle informazioni su larga scala. I dati possono essere distribuiti ma, essendo interconnessi, è possibile per un'applicazione iniziare da una

sezione di Linked Data in un sito e seguire i collegamenti incorporati verso altri dati correlati, per completare la propria base di conoscenza [17].

2.1.1 Resource Description Framework

Il *Resource Description Framework* (RDF) è un linguaggio per descrivere fatti (basato sulla sintassi di XML) sotto forma di triple ($< soggetto, predicato, oggetto >$) che possono fare riferimento a precisi vocabolari condivisi, principalmente scritti con il linguaggio RDF Schema (RDFS) [14]. Questa struttura si è rivelata essere un modo naturale per descrivere la stragrande maggioranza dei dati elaborati dalle macchine [4].

Gli elementi delle triple vengono identificati univocamente utilizzando gli *Uniform Resource Identificator* (URI). RDF utilizza gli URI per codificare in modo univoco e non ambiguo le risorse, garantendo che i concetti non siano solo stringhe in un documento, ma che siano effettivamente legati ad una definizione universale che tutti possono trovare sul Web [4].

L'insieme delle triple *soggetto*, *predicato* e *oggetto* [14] dà luogo ad un grafo RDF. Ogni tripla può essere interpretata come una proposizione che dichiara una relazione tra un soggetto e un oggetto attraverso un predicato. Formalmente, le triple sono rappresentate come segue:

- *Soggetto*: l'entità o risorsa di cui si sta parlando. Può essere un URI o un nodo anonimo.
- *Predicato*: la proprietà che definisce il tipo di relazione tra il soggetto e l'oggetto. È sempre un URI che identifica un termine in un vocabolario.
- *Oggetto*: l'entità a cui si riferisce la relazione. Può essere un URI che denota un'altra risorsa, oppure un letterale.

Oltre agli URI, quindi, le triple possono assumere come valori anche letterali e nodi anonimi. I nodi anonimi (blank nodes) vengono trattati semplicemente come indicazioni dell'esistenza di un'entità, senza utilizzare o dire nulla riguardo al nome di tale entità [14].

I letterali, invece, sono valori o stringhe di caratteri che possono essere di due

tipi principali: *letterali semplici* (plain literals) e *letterali tipizzati* (typed literals). I letterali semplici sono stringhe di caratteri che possono essere accompagnate da un tag di linguaggio, mentre i letterali tipizzati sono associati a un tipo di dato specifico, tipicamente di XML. In entrambi i casi, il letterale rappresenta un dato concreto che viene interpretato in base al suo contenuto e, nel caso dei letterali tipizzati, al tipo di dato associato [14].

2.1.2 RDF Schema

Il linguaggio RDF ha la capacità di definire dati sotto forma di triple, ma non offre la possibilità di definire schemi di dati, come classi, utilizzati anche per eseguire task di inferenza, come verificare la consistenza dei dati o derivare nuove informazioni. Per questo motivo, il W3C nel 2004 ha raccomandato il linguaggio *RDF Schema* (RDFS), un'estensione semantica di RDF che fornisce i meccanismi per descrivere gruppi di risorse correlate e le relazioni tra tali risorse [5].

RDF Schema consente la definizione di vocabolari per descrivere un insieme di risorse e le relazioni tra di esse, utilizzando principalmente i costrutti di *classi* e *proprietà*. RDFS permette di definire classi per raggruppare risorse simili e stabilire proprietà per descrivere le relazioni tra queste classi. A differenza di altri sistemi, RDFS descrive le proprietà in relazione alle classi di risorsa a cui si applicano [5].

Le *classi* in RDFS rappresentano gruppi di risorse e sono trattate come risorse stesse, spesso identificate da URI. RDFS distingue tra una classe e l'insieme delle sue istanze, conosciuto come estensione della classe. Due classi possono avere lo stesso insieme di istanze ma differire nelle loro proprietà e nella loro definizione. Ad esempio, la classe definita dall'ufficio delle imposte per le persone che vivono allo stesso indirizzo può differire dalla classe definita dall'ufficio postale per le persone con lo stesso CAP, anche se entrambe le classi potrebbero avere le stesse istanze [5].

Le classi stesse possono essere istanze di altre classi. Ad esempio, `rdfs:Class` è la classe di tutte le classi RDF, e `rdfs:Resource` è la classe di tutte le risorse descritte da RDF. Inoltre, `rdfs:Literal` rappresenta la classe dei valori letterali, come stringhe e numeri, e `rdfs:Datatype` è la classe dei tipi di dato. Le classi

possono anche essere sottoclassi di altre classi, e la proprietà `rdfs:subClassOf` è utilizzata per stabilire queste relazioni gerarchiche. Una classe può quindi essere una sottoclasse di un'altra, il che implica che tutte le istanze della sottoclasse sono anche istanze della super-classe [5]. Per esempio, nel grafo in Figura 2.2 è possibile vedere l'organizzazione gerarchica della classi, infatti `ex:cat` è sottoclasse di `ex:animal`, quindi ogni istanza di `ex:cat` sarà anche istanza di `ex:animal` [24].

Le *proprietà* in RDFS stabiliscono relazioni tra risorse e possono essere descritte attraverso diversi costrutti. Per esempio, la proprietà `rdfs:range` definisce i tipi di valore che una proprietà può assumere, specificando che i valori di una proprietà sono istanze di una o più classi. Quindi, per una proprietà p, il range di p specifica il valore che può assumere l'oggetto di una tripla con p come predicato. Analogamente, `rdfs:domain` definisce il dominio, ovvero le classi di risorse per le quali una proprietà p è applicabile. Entrambe le proprietà `rdfs:range` e `rdfs:domain` possono riferirsi a se stesse, con `rdfs:range` che ha come range la classe `rdfs:Class` e `rdfs:domain` che ha come dominio la classe `rdf:Property` [5].

Le proprietà `rdfs:subClassOf` e `rdfs:subPropertyOf` sono utilizzate per esprimere relazioni gerarchiche, rispettivamente tra classi e proprietà. Se la proprietà `rdfs:subClassOf` fa riferimento alle classi, la proprietà `rdfs:subPropertyOf` definisce una proprietà come una sotto-proprietà di un'altra. Queste proprietà sono transitive, il che implica che le relazioni si estendono lungo la gerarchia [5].

Inoltre, `rdfs:label` e `rdfs:comment` offrono modi per fornire etichette e descrizioni leggibili per le risorse, migliorando la loro comprensione e documentazione da parte degli umani [5].

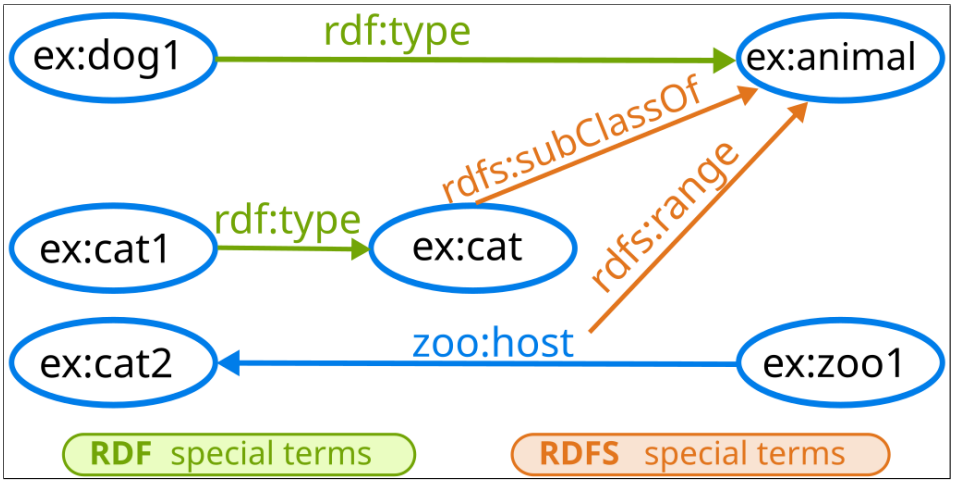

Figura 2.2: Grafo che mostra l'utlizzo di RDF e RDFS [24].

Nella Figura 2.2 è possibile osservare un esempio di grafo RDF che utilizza anche concetti di RDFS [24]. In particolare, possiamo vedere che `ex:dog1` e `ex:cat1` sono entrambe istanze della classe `ex:animal` e `ex:cat`, rispettivamente. La relazione `rdfs:subClassOf` tra `ex:cat` ed `ex:animal` indica che ogni istanza di `ex:cat` è anche un'istanza di `ex:animal`, confermando la gerarchia tra classi. Inoltre, il grafo mostra una proprietà `zoo:host` che collega `ex:cat2` a `ex:zoo1`, evidenziando l'uso di proprietà definite dall'utente. È presente anche la proprietà `rdfs:range`, che indica che il dominio di `zoo:host` appartiene alla classe `ex:zoo1`.

Tra le altre caratteristiche di RDFS, vi è l'uso di contenitori come `rdf:Bag`, `rdf:Seq` e `rdf:Alt`, che rappresentano collezioni rispettivamente non ordinate, ordinate e alternative. Inoltre, RDFS include strumenti per descrivere collezioni chiuse tramite liste (`rdf:List`), reificare dichiarazioni RDF con `rdf:Statement`, e una serie di proprietà di utilità come `rdfs:seeAlso` e `rdfs:isDefinedBy`. Queste proprietà, in particolare, sono progettate per facilitare il matching tra diversi grafi RDF, consentendo di collegare entità che, pur avendo nomi diversi, si riferiscono allo stesso concetto. Tali costrutti fanno parte del vocabolario standard di RDFS e contribuiscono ad una migliore annotazione / descrizione semantica dei dati [5].

2.1.3 SPARQL Protocol and RDF Query Language

Per interrogare dati descritti in RDF/RDFS è stato proposto dal W3C il linguaggio *SPARQL Protocol and RDF Query Language* (SPARQL). L'ultima versione, SPARQL 1.1, è raccomandazione del consorzio W3C dal 2013 [32].

Dal punto di vista formale, una query SPARQL è definita come una quadrupla (GP, DS, SM, R) in cui [23]:

- GP è un *graph pattern* (modello di grafo),
- DS è un *RDF Dataset* (insieme di dati RDF),
- SM è un insieme di *solution modifiers* (modificatori di soluzione),
- R è una *result form* (forma del risultato).

Una query SPARQL è quindi composta da quattro elementi principali. Il *graph pattern* GP specifica il modello RDF che la query cerca nei dati e può includere vari tipi di pattern, come Basic Graph Patterns, Optional o Union. DS rappresenta il dataset RDF interrogato, composto da un grafo predefinito e da eventuali grafi denominati. I *solution modifiers* SM comprendono operazioni come ordinamento, filtraggio e rimozione di duplicati, mentre la *result form* R definisce il formato del risultato, come `SELECT`, `CONSTRUCT`, `ASK` o `DESCRIBE`, permettendo di estrarre dati o costruire nuovi grafi [23].

Supponendo che i dati di un grafo di esempio, come quello presentato nella Figura 2.3, siano caricati in un servizio HTTP che può elaborare interrogazioni SPARQL, il linguaggio SPARQL 1.1 può essere utilizzato per formulare query che vanno dalla semplice corrispondenza di graph patterns fino a interrogazioni complesse [32].

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.org/alice#me> a foaf:Person .
<http://example.org/alice#me> foaf:name "Alice" .
<http://example.org/alice#me> foaf:mbox <mailto:alice@example.org> .
<http://example.org/alice#me> foaf:knows <http://example.org/bob#me> .
<http://example.org/bob#me> foaf:knows <http://example.org/alice#me> .
<http://example.org/bob#me> foaf:name "Bob" .
<http://example.org/alice#me> foaf:knows <http://example.org/charlie#me> .
<http://example.org/charlie#me> foaf:knows <http://example.org/alice#me> .
<http://example.org/charlie#me> foaf:name "Charlie" .
<http://example.org/alice#me> foaf:knows <http://example.org/snoopy> .
<http://example.org/snoopy> foaf:name "Snoopy"@en .
```

Figura 2.3: Un grafo RDF in sintassi Turtle che descrive Alice e i suoi contatti sociali [32].

A questo punto, con SPARQL 1.1 è possibile eseguire interrogazioni sul grafo RDF di tipo `SELECT`, così già possibile dalla prima versione SPARQL 1.0, ma anche nuove di tipo `ASK`, che restituiscono una risposta booleana, e `CONSTRUCT`, con le quali si possono costruire nuovi grafi RDF a partire dal risultato di una query. Questi ultimi due tipi di query sono stati aggiunti nell'ultima versione 1.1 del linguaggio SPARQL [32]. Rispetto alla 1.0, SPARQL 1.1 aggiunge una serie di nuove funzionalità al linguaggio di query, inclusi sottoquery, assegnazione di valori, espressioni di percorso o aggregati, come `COUNT` utilizzato nella Figura 2.4.

Una interrogazione in SPARQL, dal punto di vista sintattico, è composta dalle seguenti cinque parti:

- `PREFIX`: Definisce abbreviazioni per gli URI e introduce uno o più vocabolari (opzionale).
- `SELECT`, `DESCRIBE`, `ASK` e `CONSTRUCT`: Determinano il tipo di risultato che la query deve ottenere.
- `FROM` e `FROM NAMED`: Definiscono il dataset dal quale estrarre i dati (opzionale).
- `WHERE`: Contiene il pattern di triple che descrive le condizioni che i dati presenti nel grafo devono soddisfare.
- `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, `OFFSET`, `BINDINGS`: Modificatori della

query per raggruppare, filtrare, ordinare e limitare i risultati (opzionale).

La Figura 2.4 mostra una query SPARQL di tipo `SELECT` per chiedere i nomi delle persone e il numero dei loro amici a partire dal grafo RDF nella Figura 2.3 [32]. Viene utilizzato l'operatore `COUNT` per contare il numero di amici (`?friend`) per ciascuna persona, e i risultati sono raggruppati per persona (`GROUP BY ?person ?name`).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
} GROUP BY ?person ?name
```

Figura 2.4: Una query SPARQL per ottenere i nomi delle persone e il numero dei loro amici [32].

I risultati delle query `SELECT` in SPARQL consistono in insiemi di mappature da variabili a termini RDF, spesso rappresentati in forma tabellare. La Tabella 2.1 mostra in forma tabellare i risultati derivanti dalla query in Figura 2.4. Vengono indicati il numero totale di amici per ciascuna persona nel grafo RDF in Figura 2.3. Per scambiare questi risultati in una forma leggibile dalla macchina, SPARQL supporta quattro formati di scambio comuni, ovvero eXtensible Markup Language (XML), JavaScript Object Notation (JSON), Comma-Separated Values (CSV) e Tab-Separated Values (TSV) [32].

?name	**?count**
Alice	3
Bob	1
Charlie	1

Tabella 2.1: Risultati della query SPARQL in Figura 2.4 sul grafo RDF in Figura 2.3 [32].

Un aspetto avanzato dell'utilizzo di SPARQL è la possibilità di combinare le query con inferenze basate su ontologie, come quelle definite in OWL (Web Ontology Language). Le inferenze permettono di dedurre nuove triple RDF a partire dalle regole e dai dati già esistenti nel grafo. Utilizzando SPARQL insieme a un motore di

inferenza OWL, è possibile ottenere risultati che includono non solo i dati espliciti, ma anche quelli impliciti, ampliando notevolmente le capacità di interrogazione e arricchendo la semantica delle risposte.

2.1.4 Web Ontology Language

Il *Web Ontology Language* (OWL) è un linguaggio definito per rappresentare su Web la conoscenza in maniera espressiva sotto forma di ontologie. Una ontologia descrive un dominio in termini di classi, proprietà e individui e può includere descrizioni dettagliate/complesse delle caratteristiche di tali oggetti [15].

Sviluppato dal W3C, il linguaggio OWL estende RDF e RDF Schema introducendo una maggiore espressività per la definizione di classi, proprietà e vincoli. OWL è un linguaggio basato sulla logica computazionale e in particolare sulle logiche descrittive. Una *logica descrittiva* viene utilizzata per rappresentare formalmente classi, proprietà e individui [1].

Uno dei concetti fondamentali delle logiche descrittive è la separazione tra:

- *TBox* (o base di conoscenza terminologica), che descrive la terminologia e definisce il significato dei simboli;
- *ABox* (o base di conoscenza assertiva), che specifica ciò che è vero in un determinato momento.

Di solito, la base di conoscenza terminologica viene definita durante la progettazione del sistema e rappresenta l'ontologia. Essa cambia solo quando il significato del vocabolario viene modificato, un evento che dovrebbe essere relativamente raro. Al contrario, la base di conoscenza assertiva contiene informazioni specifiche a una situazione ed è nota soltanto durante l'esecuzione del sistema. La base di conoscenza assertiva viene generalmente espressa tramite triple RDF, mentre un linguaggio come OWL viene utilizzato per definire la base di conoscenza terminologica.

Quindi, OWL consente di esprimere la conoscenza in modo formale e di renderla disponibile ai programmi informatici, permettendo a essi di svolgere inferenza sulla

conoscenza[4]. Ad esempio, le ontologie OWL possono essere utilizzate per garantire la coerenza della conoscenza o per rendere esplicita la conoscenza implicita tramite servizi di inferenza, facilitando così l'interoperabilità dei dati. Tra i software più comuni per effettuare inferenza su OWL, detti *reasoners*, vi sono Hermit[5], Pellet[6], Konclude[7] e JFact[8].

La versione attuale di OWL è chiamata OWL 2 ed è stata aggiornata nel 2012. OWL 2 è diviso in tre sottolinguaggi, detti profili, ognuno dei quali bilancia l'espressività di OWL con la complessità computazionale dei servizi di inferenza. I profili OWL 2 sono OWL 2 EL, OWL 2 QL e OWL 2 RL, ciascuno progettato per ottimizzare specifici scenari applicativi, come l'efficienza nel ragionamento su grandi ontologie o l'uso diretto di triple RDF [15].

2.2 Annotazione semantica

Per poter parlare del concetto di annotazione semantica, è necessario definire formalmente il concetto di *annotazione*. Si può definire una *annotazione* A come una quadrupla (s, p, o, c), dove:

- s è il soggetto dell'annotazione (i dati annotati),
- o è l'oggetto dell'annotazione (i dati usati per annotare),
- p è il predicato (la relazione di annotazione) che definisce il tipo di relazione tra s e o,
- c è il contesto in cui viene effettuata l'annotazione [21].

A questo punto, si può proseguire con la definizione di *annotazione formale*, a seconda dei valori assunti dalla quadrupla descritta nella definizione precedente. Un'*annotazione formale* A_f è un'annotazione A, dove il soggetto s è un URI, il

[4] `https://www.w3.org/2001/sw/wiki/OWL`

[5] `https://github.com/phillord/hermit-reasoner`

[6] `https://github.com/stardog-union/pellet`

[7] `https://github.com/konclude/Konclude`

[8] `https://jfact.sourceforge.net/`

predicato p è un URI, l'oggetto o è un URI o un oggetto formale, e il contesto c è un URI. Quindi, le annotazioni formali sono leggibili dalle macchine in quanto gli elementi che ne fanno parte sono identificati da URI [21].

Si può specializzare una annotazione formale in *annotazione ontologica*, definita come segue. Un'*annotazione ontologica* A_s è un'annotazione formale A_f, dove il predicato p e il contesto c sono un termine ontologico (arbitrariamente complesso), e l'oggetto o è conforme a una definizione ontologica di p. In questo caso, gli elementi dell'annotazione non solo sono URI, ma fanno anche riferimento ad una organizzazione ontologica dei concetti [21].

Si definisce *annotazione semantica* un'annotazione che non solo utilizza URI e termini ontologici, ma che esprime significati e relazioni più complessi, espressi sotto forma di triple RDF. In questo modo, si consente alle macchine di interpretare e utilizzare i dati in modo più significativo. Le annotazioni semantiche migliorano la capacità di recupero e integrazione delle informazioni, facilitando la comprensione del contenuto e del contesto da parte di applicazioni automatizzate.

Quindi, per rendere disponibili e interpretabili dalle macchine i contenuti di una pagina Web è necessario che siano annotati in modo formale sotto forma di triple. Storicamente, il contenuto delle pagine nel Web veniva pubblicato unicamente e interamente in linguaggio HTML, progettato soprattutto per la presentazione delle informazioni. Pertanto, i programmi e i servizi che necessitavano l'accesso a dati strutturati dovevano estrapolarli direttamente dal codice HTML. Questo però causava errori di varia natura, oltre ad essere necessaria una procedura diversa per ogni sito Web [12].

Le annotazioni semantiche risolvono questo problema, facendo uso di formati di marcatura standard con una sintassi specifica per annotare gli elementi in una pagina HTML. Per assicurare che i dati possano essere interpretati in modo uniforme, il significato semantico è codificato all'interno di ontologie e vocabolari, strutturati e condivisi. In passato, le annotazioni semantiche erano spesso disgiunte e facevano riferimento a diversi approcci e vocabolari, il che rendeva difficile l'integrazione e l'interoperabilità dei dati. Questa mancanza di standardizzazione limitava le potenzialità delle annotazioni semantiche nel promuovere una comprensione coerente e unificata del contenuto Web [21]. Tuttavia, l'introduzione di vocabolari come

Schema.org ha rappresentato un passo significativo verso l'integrazione e la condivisione di dati semantici, facilitando l'adozione di pratiche di annotazione più uniformi e interoperabili [12].

2.2.1 Schema.org

Tra il 1997 e il 2004 sono stati sviluppati diversi standard, come RDF, RDF Schema e OWL, per definire la sintassi e il modello dei dati. Ad accompagnare questi standard, sono stati proposti una serie di vocabolari condivisi relativi a molteplici settori come RSS per i feed di notizie, vCard per lo scambio di informazioni di contatto, hCalendar per i dati sui calendari e molti altri [12].

Questi vocabolari, essendo diversi e indipendenti, non erano universali e compatibili l'un l'altro, riducendo di gran lunga le potenzialità del Web semantico, specialmente per quanto riguarda la ricerca sul Web. Per migliorare la ricerca era necessario utilizzare *parser*[9], spesso complessi, per estrarre le informazioni strutturate, ma erano fragili e non portarono ai progressi attesi. La necessità di avere un vocabolario condiviso, applicabile in molti contesti del Web, confluì nella creazione di Schema.org [12].

Schema.org è un vocabolario condiviso progettato per migliorare la rappresentazione semantica di dati strutturati in una vasta gamma di domini aplicativi. Esso copre ambiti di applicazione come prodotti, eventi, recensioni, articoli, contenuti multimediali, organizzazioni, persone e molti altri, facilitando l'interoperabilità tra i motori di ricerca e le applicazioni Web. Gli schemi di Schema.org sono estensibili e definiti in modo tale da essere comprensibili sia dai motori di ricerca che da altre applicazioni, aiutando a strutturare meglio le informazioni [26].

Schema.org è un progetto collaborativo, avviato nel 2011 da Google, Microsoft, Yahoo e Yandex, che ha coinvolto anche la comunità scienfifica e membri del W3C, con l'obiettivo comune di fornire un insieme condiviso di schemi di dati strutturati.

[9]Modulo software che analizza dati di input, tipicamente testuali, creando una struttura dati che rappresenta la sintassi di quel testo. Durante il parsing, il parser verifica la correttezza sintattica secondo le regole grammaticali, facilitando l'elaborazione successiva da parte di compilatori o interpreti.

Questo vocabolario semantico è definito come un insieme di schemi estensibili di dati strutturati, semplificando il compito degli sviluppatori Web grazie a un'unica struttura ampiamente utilizzata da motori di ricerca e molte altre applicazioni [26].

Un aspetto fondamentale dello sviluppo di Schema.org riguarda le decisioni di progettazione adottate con l'obiettivo di facilitare l'uso del vocabolario da parte degli amministratori dei siti Web, spostando il carico maggiore sui consumatori di dati strutturati (ad esempio, motori di ricerca o altre applicazioni che elaborano i dati strutturati). È per questo motivo che dal punto di vista della sintassi, Schema.org non accetta un unico formato di marcatura ma diversi, quali *RDFa*, *Microdata* e *JSON-LD* [12].

Infatti, ogni formato ha caratteristiche peculiari e sono tutti utilizzati, in modo da bilanciare la flessibilità per i publisher e le esigenze delle applicazioni. Microdata è stato concepito per gestire la complessità di RDFa 1.0, mentre JSON-LD ha guadagnato popolarità grazie alla sua capacità di integrarsi facilmente con applicazioni basate su JavaScript lato client e strumenti come i widget incorporati in siti esterni. La scelta di adattare Schema.org a più sintassi si è dimostrata efficace, poiché lascia agli sviluppatori la libertà di adottare il formato che si adatta meglio alla propria piattaforma [12].

Schema.org facilita la creazione di una base di conoscenza coerente e integrata, utilizzando un vocabolario comune per connettere e arricchire informazioni da fonti diverse, come possibile vedere in Figura 2.5. L'immagine illustra una base di conoscenza, rappresentata come grafo, costruita integrando dati provenienti da più fonti che usano dati strutturati con Schema.org. È rappresentato un *grafo aciclico orientato* (DAG) in cui i nodi corrispondono a entità, caratterizzate dai campi `@type` e `@id`, o valori testuali. Gli archi indicano invece le relazioni, stilizzate in modi diversi per riflettere le fonti dei dati differenti. Ad esempio, "Under the Pink" e "Crucify" sono entrambi in relazione `instanceof` con "music album" e in relazione `author` a "Tori Amos". Quest'ultima è a sua volta collegata con relazione `birthplace` a "Newton, NC" a cui possono essere collegate informazioni aggiuntive con la relazioni `located in`, `instanceof` e `temperature` [12].

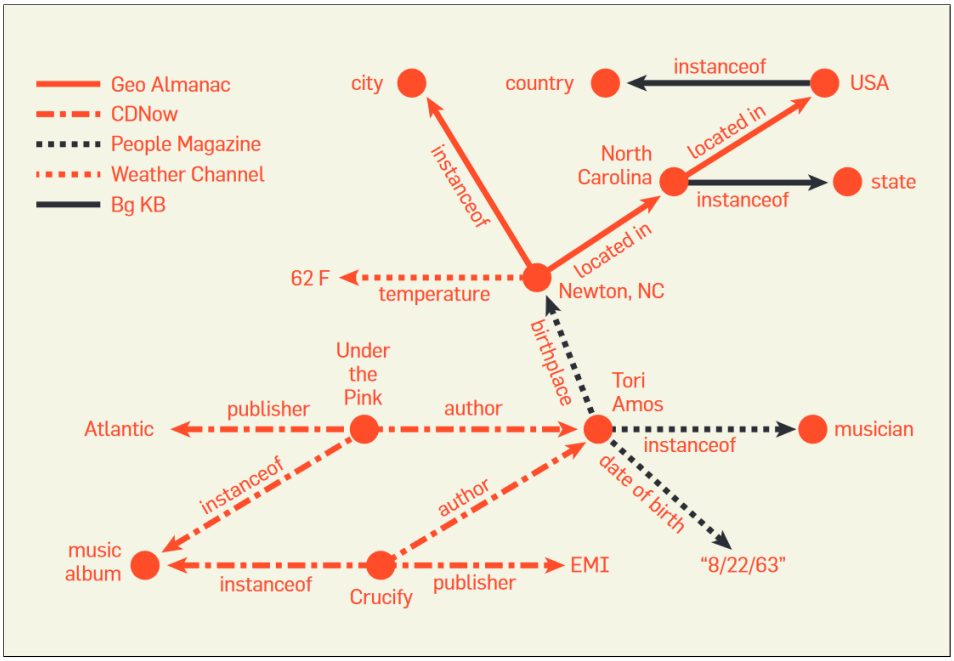

Figura 2.5: Esempio di base di conoscenza derivata da più siti che usano Schema.org, sotto forma di grafo [12].

Una caratteristica di Schema.org è di essere sempre in espansione in base alle necessità dei suoi utilizzatori. Fu lanciato nel 2011 con 297 classi e 187 relazioni, cresciute nel 2016 a 638 classi e 965 relazioni [12], fino ad arrivare alle 811 classi e 1484 relazioni attuali[10]. Le classi sono organizzate in una gerarchia, dove ogni classe può avere una o più superclassi, anche se la maggior parte ne ha solo una. Le relazioni sono polimorfiche nel senso che hanno uno o più domini e uno o più range e questa è una differenza sostanziale con linguaggi come RDF Schema e OWL che prevedono un singolo dominio e range per ogni relazione, giustificata con la possibilità di riutilizzare relazioni esistenti senza modificare significativamente la gerarchia delle classi [12].

2.2.2 Resource Description Framework in Attributes

Resource Description Framework in Attributes (RDFa) è un formato standard di annotazione che può essere utilizzato con il vocabolario condiviso Schema.org. RDFa

[10] `https://schema.org/docs/schemas.html`

è un'estensione di HTML5 che fornisce un insieme di attributi HTML per arricchire i dati visivi con indicazioni leggibili dalle macchine. Utilizzando RDFa, gli autori possono trasformare il testo e i link visibili dagli esseri umani in dati leggibili dalle macchine, senza dover ripetere i contenuti [29]. La versione attuale di RDFa è la 1.1 e rappresenta una sostanziale semplificazione rispetto alla complessità della versione precedente, RDFa 1.0 [12].

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Example Document</title>
  </head>
  <body vocab="http://schema.org/">
    <p typeof="Blog">
      Welcome to my <a property="url" href="http://example.org/">blog</a>.
    </p>
  </body>
</html>
```

Figura 2.6: Esempio di documento HTML+RDFa 1.1 [29].

I dati strutturati vengono incorporati direttamente all'interno del corpo della pagina Web, nella stessa posizione in cui si trovano gli elementi visibili. Un semplice esempio di documento HTML con RDFa è presentato nella Figura 2.6. Come è possibile notare, l'elemento `<p>` è contrassegnato con l'attributo `typeof="Blog"`, indicando che il contenuto di questo paragrafo rappresenta un blog secondo il vocabolario di Schema.org. Inoltre, il link `<a>` è arricchito con l'attributo `property="url"`, che specifica che l'URL del blog è `http://example.org/`.

Le informazioni derivanti dalle annotazioni dell'esempio in Figura 2.6 possono essere estratte da un processore RDFa conforme allo standard del W3C ed essere mostrate in formato Turtle, che permette ad un grafo RDF di essere scritto in forma testuale. Le informazioni strutturate in formato Turtle potranno essere poi elaborate dalle applicazioni, come ad esempio i motori di ricerca, per comprendere il significato semantico dei dati e agirne di conseguenza [29].

Il codice in Figura 2.7 rappresenta la traduzione in formato Turtle delle annotazioni RDFa nel documento HTML in Figura 2.6, sotto forma di tripla RDF. La tripla specifica che l'elemento è di tipo `Blog` e associa l'URL `http://example.org/`

al predicato `url`. In questo caso, l'oggetto è un letterale che rappresenta l'URL del blog, mentre il soggetto e il predicato sono identificati tramite URI del vocabolario Schema.org [29].

```
[] a <http://schema.org/Blog>;
   <http://schema.org/url> <http://example.org/> .
```

Figura 2.7: Tripla RDF estratta dall'esempio in Figura 2.6 [29].

2.2.3 Microdata

Il formato *Microdata* è stato sviluppato come parte di HTML5 con l'obiettivo di fornire un modo più semplice per annotare gli elementi HTML con etichette leggibili dalle macchine rispetto a RDFa 1.0, decisamente più complesso [12]. Come RDFa, anche Microdata prevede di incorporare le informazioni strutturate direttamente all'interno del *body* della pagina Web, arricchendo di significato semantico direttamente gli elementi visibili nella pagina [25].

Utilizzando Microdata, è possibile definire tipi di elementi e proprietà direttamente nei tag HTML tramite tre attributi principali: `itemscope`, `itemtype` e `itemprop`. L'attributo `itemscope` definisce un contesto per l'annotazione di un tipo di elemento, mentre `itemtype` specifica il tipo di elemento, utilizzando un URL che punta a un vocabolario come Schema.org. L'attributo `itemprop` è utilizzato per identificare le proprietà dell'elemento specificato [25].

```
<div itemscope itemtype="https://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <span>Director: <span itemprop="director">James Cameron</span> (born August
  ↪  16, 1954)</span>
  <span itemprop="genre">Science fiction</span>
  <a href="../movies/avatar-theatrical-trailer.html"
  ↪  itemprop="trailer">Trailer</a>
</div>
```

Figura 2.8: Esempio di annotazione di una pagina HTML sul film Avatar usando Microdata [25].

Nella Figura 2.8 è proposto un esempio di utilizzo del formato Microdata in una pagina HTML riguardante il film Avatar. Viene utilizzato l'attributo `itemscope`

per identificare il contenuto della pagina riguardante un particolare elemento, nel presente caso il film Avatar. Con l'attributo `itemtype` viene specificato il tipo dell'elemento usando l'URL del vocabolario Schema.org, in questo caso il tipo è `https://schema.org/Movie`. All'interno di questo contesto, l'attributo `itemprop` è utilizzato per annotare le singole proprietà dell'elemento, come il nome del film (`name`), il regista (`director`), il genere (`genre`) e il link al trailer (`trailer`) [25].

Rispetto a RDFa, Microdata presenta alcune differenze formali e di implementazione. RDFa tende ad essere più complesso e flessibile, consentendo una maggiore personalizzazione attraverso attributi come `typeof`, `property` e `resource`, e può essere utilizzato in combinazione con vari vocabolari. Inoltre, RDFa è un formato standard per la serializzazione delle triple RDF [29].

Microdata, invece, nasce con l'obiettivo di fornire una sintassi più semplice e diretta per annotare i contenuti HTML, integrandosi in HTML5 tramite un insieme limitato di attributi: `itemscope`, `itemtype`, e `itemprop`. Questi attributi, sebbene siano strettamente definiti e facili da utilizzare, permettendo agli autori di annotare i contenuti con gruppi di coppie nome-valore in modo coerente, non permettono una grande flessibilità. A differenza di RDFa, che è progettato per essere utilizzato con diversi vocabolari di ontologie sulla stessa pagina Web, Microdata è più restrittivo in questo aspetto. Microdata tende a focalizzarsi su un vocabolario alla volta per ogni elemento annotato, rendendo più difficile l'integrazione di più vocabolari nella stessa pagina [33].

2.2.4 JavaScript Object Notation for Linked Data

Il formato *JavaScript Object Notation for Linked Data*, abbreviato JSON-LD, è l'ultimo formato di annotazione di dati strutturati adottato da Schema.org, dove i dati strutturati sono rappresentati come un insieme di oggetti in stile JavaScript, separati dal corpo del documento HTML e incorporati in un tag `<script>`. È un formato particolarmente adatto per i siti che vengono generati utilizzando JavaScript lato client, così come per le email personalizzate dove le strutture dei dati possono essere notevolmente più elaborate [12].

JSON-LD è definito dal W3C come una sintassi leggera per serializzare Linked

Data utilizzando il linguaggio JSON. È stato principalmente concepito per l'utilizzo di Linked Data negli ambienti di programmazione Web, al fine di costruire servizi Web interoperabili e memorizzare Linked Data in motori di archiviazione basati su JSON [17].

JSON-LD è stato progettato per essere pienamente compatibile con JSON, in modo tale da riutilizzare il vasto numero di parser e librerie JSON attualmente disponibili. Siccome è un formato di annotazione semantica, può essere utilizzato come RDF insieme ad altre tencnologie di Linked Data come SPARQL. L'ultima versione del formato è JSON-LD 1.1 ed è raccomandazione del W3C dal 2020 [17].

Oltre alle caratteristiche offerte da JSON, JSON-LD introduce [17]:

- un meccanismo di identificazione universale di oggetti JSON tramite l'uso di Internationalized Resource Identifiers (IRI),
- un modo per disambiguare chiavi comuni in documenti JSON diversi associandole ad IRI tramite un contesto definito attraverso il tag `@context`,
- un meccanismo in cui un valore in un oggetto JSON può riferirsi a una risorsa su un sito Web diverso utilizzando il tag `@id`,
- la possibilità di annotare stringhe con la relativa lingua con il tag `@language`,
- un modo per associare tipi di dati con valori, come date e orari, specificando il tipo di dati tramite il tag `@type`,
- una funzionalità per descrivere uno o più grafi orientati in un unico documento JSON-LD utilizzando il tag `@graph`.

Nella Figura 2.9 è possibile vedere un esempio di dati strutturati JSON-LD inseriti all'interno di una pagina HTML. Come è possibile notare, i dati JSON-LD sono separati dal corpo della pagina e vengono incapsulati all'interno di un tag `<script>`, con il tipo `application/ld+json`. In particolare, il codice in Figura 2.9 descrive una ricetta di una torta, comprensiva di dettagli come il nome della ricetta, l'autore e altre informazioni pertinenti [11].

Come è possibile notare, i tag sono utilizzati per definire le informazioni strutturate. Il tag `@context` indica che lo schema utilizzato per descrivere i dati appartiene

al vocabolario di `schema.org` mentre il tag `@type` specifica il tipo di entità descritta, in questo caso `Recipe`, che identifica il documento come una ricetta. Il campo `name` contiene il nome della ricetta, "Party Coffee Cake". Il tag `author` utilizza a sua volta un oggetto con il tipo `Person`, per specificare l'autore della ricetta, che in questo caso è `Mary Stone`. Successivamente, `datePublished` rappresenta la data di pubblicazione della ricetta nel formato ISO 8601[11], e `description` fornisce una breve descrizione della ricetta. Infine, il tag `prepTime` utilizza il formato `PT20M` per specificare il tempo di preparazione della ricetta [11].

```
<html>
  <head>
    <title>Party Coffee Cake</title>
    <script type="application/ld+json">
    {
      "@context": "https://schema.org/",
      "@type": "Recipe",
      "name": "Party Coffee Cake",
      "author": {
        "@type": "Person",
        "name": "Mary Stone"
      },
      "datePublished": "2018-03-10",
      "description": "This coffee cake is awesome and perfect for parties.",
      "prepTime": "PT20M"
    }
    </script>
  </head>
  <body>
    <h2>Party coffee cake recipe</h2>
    <p>
      <i>by Mary Stone, 2018-03-10</i>
    </p>
    <p>
      This coffee cake is awesome and perfect for parties.
    </p>
    <p>
      Preparation time: 20 minutes
    </p>
  </body>
</html>
```

Figura 2.9: Esempio di dati strutturati JSON-LD incorporati in una pagina HTML per descrivere una ricetta [11].

[11] `https://www.iso.org/iso-8601-date-and-time-format.html`

Il formato JSON-LD viene attualmente consigliato per la Ricerca Google, rispetto a RDFa e Microdata, in quanto è il più facilmente implementabile e manutenibile su larga scala. Il *crawler*[12] di Google legge i dati strutturati nelle pagine Web per abilitare funzionalità e miglioramenti speciali dei risultati di ricerca. Ad esempio, i dati strutturati in un codice come quello in Figura 2.9 possono essere visualizzati in modo grafico in un risultato di ricerca [11].

Una pagina Web implementata con dati strutturati di un formato valido può infatti essere visualizzata dai motori di ricerca in formato grafico. La Figura 2.10 mostra come delle pagine di ricette, simili a quella definita dal codice in Figura 2.9, possono essere visualizzate in una pagina dei risultati. Il codice corrispondente al primo risultato della Figura 2.10 include diversi campi utilizzati da Google direttamente per la visualizzazione dei risultati, come `aggregateRating` per i dati sulle recensioni e sulla valutazione media, `totalTime` per il tempo totale necessario, `nutrition` per le informazioni nutrizionali come le calorie [11].

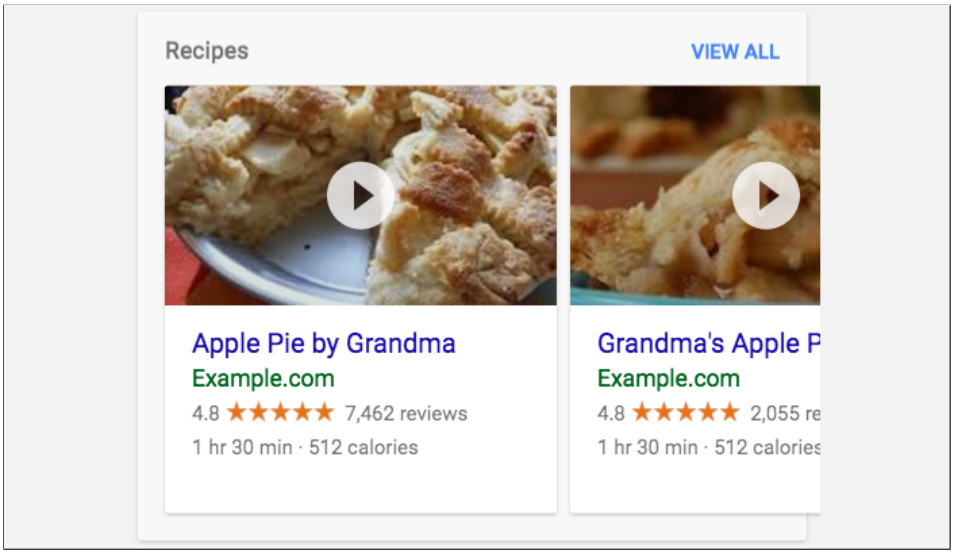

Figura 2.10: Esempio di visualizzazione grafica di un risultato con dati strutturati [11].

[12]Software che esplora le pagine Web per raccogliere le informazioni e indicizzare i contenuti per i motori di ricerca.

Capitolo 3

Framework per l'Annotazione semantica e la Valutazione

Nel presente capitolo viene fornita una descrizione dettagliata di un framework per l'annotazione semantica di risorse Web e per la valutazione dell'impatto sul ritrovamento di tali risorse da parte dei motori di ricerca. Si inizia con una panoramica generale del framework, composto dalle fasi di annotazione, raffinamento e valutazione, per poi approfondire ciascuna di queste fasi nel dettaglio.

Successivamente, si descrive la generazione delle query per la fase di valutazione, che compongono la *ground truth*, nonché la loro conversione in query testuali e semantiche. Quindi, si introducono le due strategie di ricerca oggetto di confronto: la ricerca basata su parole chiave e la ricerca semantica. Infine, si procede con una illustrazione delle metriche di valutazione utilizzate per confrontare quantitativamente i risultati restituiti dai due sistemi.

3.1 Descrizione del framework

Al fine di valutare le opportunità derivanti dall'utilizzo dell'annotazione semantica in un sito Web, Google misura quantitativamente una serie di metriche come il

click-through rate (CTR)[1], il numero di errori di indicizzazione o il traffico del sito Web, prima dell'introduzione dei dati strutturati all'interno del sito[2]. È necessario quindi aspettare da alcune settimane a diversi mesi affinché il motore di ricerca e i relativi indici siano effettivamente aggiornati, per poi eseguire nuovamente la valutazione delle metriche quantitative, e svolgere la valutazione dei risultati [8].

Per il presente lavoro di tesi, è stato necessario progettare un framework che consenta di effettuare una valutazione robusta e riproducibile dell'impatto delle annotazioni semantiche su un sito Web, con particolare attenzione all'utilizzo dei dati strutturati nel recupero delle informazioni da parte dei sistemi di ricerca. In particolare, il framework utilizzato permette di misurare l'impatto dell'annotazione semantica in un sito Web anche se non in produzione, essendo impiegabile anche su una versione in locale del sito, come viene svolto nel Capitolo 4.

Il framework è costituito da tre fasi: *annotazione*, *raffinamento* e *valutazione*. Le fasi si eseguono in sequenza come mostrato in Figura 3.1.

Figura 3.1: Framework per l'annotazione semantica di risorse Web e valutazione dell'impatto sul ritrovamento da parte dei motori di ricerca

Durante la fase di *annotazione*, il sito è sottoposto alla generazione dei dati strutturati. La scelta del formato di marcatura da utilizzare, principalmente tra i formati RDFa, Microdata e JSON-LD, dipende dalla tipologia del sito e dall'ambiente in cui opera [12]. Quindi, si esplorano le diverse tecnologie per poter effettivamente generare i dati strutturati all'interno delle pagine HTML del sito Web [10].

[1]Il click-through rate (CTR) è il rapporto tra il numero di clic su una pagina e il numero di volte in cui la pagina viene presentata tra i risultati.

[2]`https://developers.google.com/search/case-studies/overview?hl=en`

La successiva fase di *raffinamento* consiste nel miglioramento delle annotazioni semantiche create dalla fase precedente. Prevede una verifica iniziale della correttezza delle annotazioni attraverso strumenti di convalida del markup e quindi l'integrazione di eventuali dati mancanti. Successivamente, possono seguire fasi di normalizzazione dei dati, per uniformare le rappresentazioni dei dati, e di rimozione di dati ridondanti o non pertinenti. Alla fine della fase di raffinamento, i dati strutturati sono pronti a essere sottoposti ai sistemi di ricerca.

La fase finale del framework è la fase di *valutazione*. L'obiettivo di questa fase è produrre una valutazione quantitativa delle prestazioni derivanti dall'utilizzo di dati strutturati per la ricerca di elementi. Per fare una valutazione dell'impatto dell'introduzione di annotazioni semantiche nel sito, si confrontano le prestazioni di un sistema di ricerca basato su parole chiave, che non utilizza i dati strutturati per la ricerca, con un sistema di ricerca semantico basato su query SPARQL.

3.1.1 Annotazione semantica delle risorse

La fase di *annotazione* inizia con una valutazione dei formati di marcatura disponibili, che potranno essere utilizzati per annotare gli elementi all'interno delle pagine Web del sito sottoposto al framework. A seconda della tipologia del sito e dell'ambiente in cui opera, può essere più appropriato scegliere un formato di marcatura oppure un altro [12].

Diverse sintassi sono adatte a diversi strumenti e modelli di creazione dei contenuti. Generalmente, il formato JSON-LD è particolarmente adatto a siti generati utilizzando JavaScript lato client, mentre i formati Microdata e RDFa spesso funzionano meglio per siti generati utilizzando template lato server [12]. Per esempio, anche se la Ricerca Google supporta tutti e tre i formati di annotazione, purché siano implementati correttamente e in linea con la documentazione, JSON-LD è la scelta consigliata per i dati strutturati, in quanto rappresenta la soluzione più semplice per i proprietari di siti da implementare e mantenere su larga scala, risultando quindi meno soggetta a errori [11].

Dopo aver selezionato il formato di annotazione preferito per il sito Web, è necessario effettivamente generare i dati strutturati in tale formato. Sono presenti

diversi strumenti che generano i dati strutturati, in base al formato scelto.

Uno strumento è ad esempio Google Tag Manager[3], una piattaforma che consente di gestire i tag su un sito Web senza modificarne il codice. Lo strumento permette di generare dati strutturati in JSON-LD, prelevando direttamente le variabili e inserendole all'interno del tag `<script>`, per ridurre il rischio di discrepanze tra il contenuto della pagina e i dati strutturati. Un esempio di dati generati dinamicamente in base alle variabili di una pagina di ricette è presentato nel codice in Figura 3.2 [10]. Questo sarà il metodo utilizzato per generare i dati strutturati nel sito Web del caso di studio, al Capitolo 4. Un altro modo per generare dati strutturati in JSON-LD è utilizzare JavaScript per generare tutti i dati strutturati o aggiungere più informazioni ai dati strutturati renderizzati lato server [10].

```
<script type="application/ld+json">
  {
    "@context": "https://schema.org/",
    "@type": "Recipe",
    "name": "{{recipe_name}}",
    "image": [ "{{recipe_image}}" ],
    "author": {
      "@type": "Person",
      "name": "{{recipe_author}}"
    }
  }
</script>
```

Figura 3.2: Codice per generare dinamicamente i dati strutturati in JSON-LD di una pagina Web di ricette, usando le variabili [10].

Per la generazione di annotazioni in Microdata e RDFa, sono presenti diversi strumenti. Alcuni *Content Management System* (CMS) offrono supporto nativo per l'inclusione di dati strutturati in RDFa nei loro template.

Per esempio, Drupal[4] integra RDFa generando automaticamente i markup necessari all'interno dei documenti HTML. Utilizzando i template del sistema, Drupal consente di specificare il DOCTYPE e definire i namespace richiesti, assicurando

[3] `https://tagmanager.google.com/`

[4] Piattaforma software open-source CMS, implementata in linguaggio PHP: `https://www.drupal.org/`

che i dati strutturati siano correttamente formattati[5]. Anche per Microdata sono presenti strumenti analoghi, come Joomla[6], che permette di aggiungere semantiche Microdata dinamicamente al contenuto del sito, grazie alla gestione automatica dei tipi e delle proprietà definiti in Schema.org [7].

3.1.2 Raffinamento delle annotazioni

Ottenuti i dati strutturati, indipendentemente dal formato, è possibile passare alla fase di *raffinamento*, in cui si migliorano i dati strutturati interni al sito. Questa fase è necessaria per assicurarsi che tutte le informazioni annotate semanticamente siano complete e corrette, prima di poterne effettivamente valutare gli eventuali benefici.

La fase inizia con la validazione dei dati strutturati utilizzando strumenti appositi di convalida, che analizzano i dati strutturati interni alla pagina e verificano che non ci siano errori critici, errori secondari o informazioni mancanti. Gli strumenti di validazione principali sono lo *strumento di convalida* di Schema.org e il *test dei risultati avanzati* di Google. Entrambi gli strumenti di validazione del markup prendono in input URL o snippet di codice e restituiscono eventuali problemi rilevati.

Lo strumento di convalida del markup di Schema.org[8] svolge una convalida generale su tutti i dati strutturati basati su Schema.org, ma senza far riferimento a funzionalità specifiche di Google. Il test dei risultati avanzati di Google[9], invece, fa un test sui dati strutturati della pagina con l'obiettivo di verificare quali risultati arricchiti di Google possono essere generati a partire dai dati nella pagina, i *rich snippets*, nel contesto della Ricerca Google. Google divide i problemi sugli elementi annotati in critici o non critici. Gli elementi con problemi critici non sono validi,

[5] `https://www.drupal.org/node/778988`

[6] Anche Joomla è una piattaforma software open-source CMS: `https://www.joomla.org/`

[7] `https://docs.joomla.org/Microdata`

[8] `https://validator.schema.org/`

[9] `https://search.google.com/test/rich-results`

mentre quelli con problemi non critici possono essere validi, ma potrebbero essere migliorati per mostrare più informazioni su Google o per corrispondere a più query.

A seconda dei risultati ottenuti dagli strumenti di convalida, possono seguire fasi di normalizzazione dei dati, rimozione di dati ridondanti o non pertinenti, e di integrazione di dati mancanti.

Con *normalizzazione dei dati* si intende il processo di uniformare e standardizzare le informazioni per garantire che i dati siano coerenti e compatibili. La standardizzazione può essere, per esempio, dei formati per date, unità di misura o categorie di elementi, ed è fondamentale affinché i dati siano omogenei e, quindi, facilmente interpretabili dai motori di ricerca. Un esempio potrebbe essere la proprietà `"schema:prepTime"`, che può essere usata per indicare il tempo necessario di preparazione di un elemento. Consultando i file di definizione del vocabolario di Schema.org[10], il formato del tempo richiesto da questa proprietà deve aderire allo standard ISO 8601[11], progettato per gestire in modo universale e non ambiguo le date e le quantità di tempo.

Altre operazioni sui dati strutturati prevedono la rimozione di dati ridondanti o non pertinenti. Un *dato ridondante* si riferisce a informazioni duplicate o superflue che non aggiungono valore alle annotazioni esistenti. La presenza di dati ridondanti può confondere i motori di ricerca e influenzare negativamente la qualità delle informazioni presentate. Inoltre, è importante assicurarsi che tutte le informazioni siano pertinenti al contesto del sito e alle potenziali query degli utenti.

Infine, gli strumenti di convalida delle annotazioni possono suggerire l'inserimento di una serie di *dati mancanti*, che potrebbero essere utili a seconda del dominio di applicazione del sito Web. Ad esempio, nel sito Web applicato al framework per il caso di studio, il test dei risultati avanzati di Google ha suggerito l'inserimento di dati mancanti sulle recensioni e sulla valutazione dei prodotti.

A questo punto, la fase di raffinamento è completa e i dati strutturati possono essere sottoposti alla fase di valutazione.

---

[10] `https://schema.org/docs/developers.html`

[11] `https://www.iso.org/iso-8601-date-and-time-format.html`

3.1.3 Valutazione dell'impatto

La fase di *valutazione* del framework ha l'obiettivo di fornire una misura quantitativa dell'impatto derivante dall'introduzione nel sito Web dei dati strutturati generati nelle fasi precedenti del framework. È fondamentale svolgere questa fase per determinare se l'implementazione dei dati strutturati ha effettivamente avuto riscontri positivi sulla capacità del sito di proporre all'utente risultati accurati, a seconda delle interrogazioni possibili.

Sebbene l'impiego di log di query reali potrebbe riflettere più organicamente le esigenze degli utenti, questa opzione non è sempre praticabile a causa di limitazioni operative o di accesso ai dati. Pertanto, è stata creata una *ground truth* composta da query di base e risultati esatti associati. Questa metodologia, pur introducendo potenziali forme di bias, consente di produrre metriche quantitative in termini di *precisione* e *richiamo*, garantendo risultati riproducibili e prontamente disponibili. In questo modo, si effettua una valutazione preliminare della qualità dei risultati e sul confronto tra i sistemi di ricerca basati su parole chiave e quelli basati su SPARQL. Dopo la fase di valutazione proposta nel framework, è comunque possibile sottoporre il sistema al feedback degli utenti.

La *ground truth*, o verità di base, è costituita da un insieme di query associate ai loro risultati noti a priori. Avendo un insieme di query su cui eseguire la valutazione, le stesse query potranno essere tradotte in query testuali e query in SPARQL. Di conseguenza, per ogni query si produrrà un insieme di risultati derivanti da una ricerca non basata sull'utilizzo di dati strutturati e un insieme di risultati derivanti dall'esecuzione della query SPARQL su un grafo di triple RDF. I due motori di ricerca si confrontano, pertanto, con la ground truth per misurare le loro prestazioni.

Come approccio di ricerca non basato su dati strutturati è stata scelta una ricerca basata sulla corrispondenza di parole chiave a partire da una query testuale, descritta successivamente. I risultati di tale approccio derivano dall'esecuzione di query basate su parole chiave su un *catalogo di elementi*, che rappresenta l'insieme degli elementi su cui viene svolta la ricerca, solitamente collegato a un database. Il catalogo di elementi varia a seconda del dominio applicativo del sito: ad esempio, in un sito di eCommerce, gli elementi del catalogo sono i prodotti.

Sono necessari, per il confronto, anche i risultati derivanti dalla ricerca che utilizza i dati strutturati. Viene eseguita una ricerca basata su query SPARQL su un grafo di triple RDF che descrivono gli stessi elementi del catalogo su cui viene svolta la ricerca basata su parole chiave, in modo tale da garantire che i risultati siano confrontabili. Infatti, vengono eseguite le stesse query sugli stessi elementi, l'unica differenza risiede nella rappresentazione dei dati e delle query, nonché nel metodo di ritrovamento dei risultati.

Dopo aver ottenuto i risultati, è possibile produrre le metriche di valutazione. Si valutano i due approcci basandosi su dati quantitativi che misurano la bontà dei risultati prodotti. Le principali metriche di valutazione proposte nel framework sono la *precisione* e il *richiamo*, e sono calcolate confrontando i risultati dei due approcci con la ground truth.

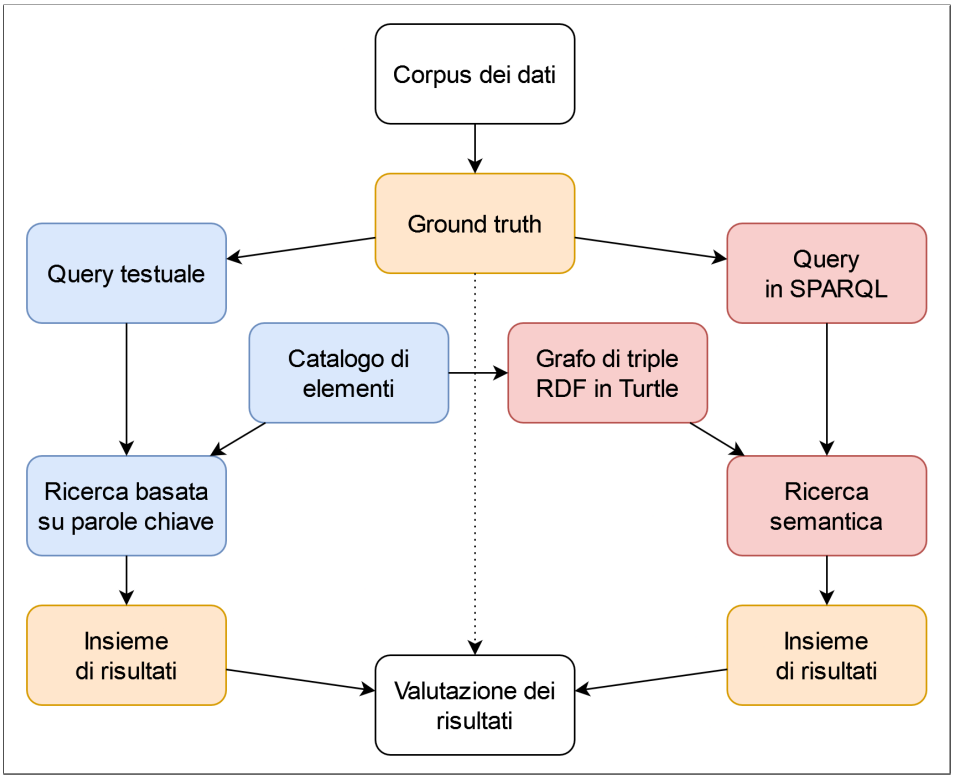

Figura 3.3: Schema che illustra la fase di valutazione del framework, con blocchi colorati per le diverse operazioni. Per evidenziare le analogie e le differenze nei due approcci, i blocchi relativi alla ricerca semantica sono in rosso, quelli dell'approccio basato su parole chiave in blu. I blocchi per il calcolo dei risultati in verde, che confrontano i risultati dei due sistemi con la ground truth calcolata a priori.

La Figura 3.1 schematizza la fase di valutazione del framework proposto e applicato in questa tesi. Il punto di partenza della fase di valutazione è costituito da un corpus dei dati del dominio. Da esso si calcolano, mediante una combinazione di caratteristiche degli elementi, le query di base. A ogni query di base è associato un insieme di elementi, che rappresenta i risultati esatti della query ed è noto a priori.

A partire dalle query di base, si calcola l'insieme delle query testuali. Ogni query testuale è direttamente utilizzata, insieme al catalogo di elementi, come input per gli algoritmi che svolgono la ricerca basata su parole chiave, che cercano una corrispondenza tra i termini che compongono la query e i termini che descrivono il prodotto. A seconda del sito Web impiegato nel framework, gli algoritmi di ricerca possono essere sviluppati appositamente per la fase di valutazione oppure possono essere inclusi in plugin già esistenti. Nel caso di studio, viene utilizzato un plugin esistente per eseguire la ricerca basata su parole chiave internamente al sito, ricercando i risultati nel catalogo di prodotti.

Viene quindi restituito un insieme di elementi in cui ogni parola chiave della query testuale trova una corrispondenza nei termini utilizzati per descrivere gli elementi. Questo significa che ciascun termine della query deve essere presente nelle descrizioni associate agli elementi affinché questi siano inclusi tra i risultati.

Le query di base sono anche convertite in query SPARQL. Viene considerato SPARQL come linguaggio di interrogazione per via della sua capacità di ricercare dati in formato RDF, che rappresentano una delle principali conseguenze dell'annotazione semantica. SPARQL consente di interrogare il grafo di triple generato dai dati semantici, sfruttando relazioni e proprietà tra gli elementi, cosa che non sarebbe possibile con semplici query basate su parole chiave.

A seconda del formato di annotazione dei dati strutturati, si possono svolgere diverse operazioni di estrazione dei dati strutturati interni alle pagine, per poterli utilizzare direttamente nella ricerca semantica. Possono essere utilizzati parser specifici per estrarre i dati semantici in base al loro formato, che sia RDFa[12], Microdata o JSON-LD. Ad esempio, nel caso di studio è stato sviluppato un codice che

[12]Un esempio di parser per RDFa è presente a questo link: `https://github.com/shellac/java-rdfa`

ricerca direttamente il campo `<script>` all'interno delle pagine Web degli elementi nel catalogo e ne estrae le informazioni in un unico file JSON-LD.

Estratti i dati strutturati, essi possono essere trasformati in un grafo di triple RDF, serializzato in linguaggio Turtle, che contiene tutte le informazioni sugli elementi annotate nelle fasi precedenti. A questo punto, ogni query SPARQL e l'insieme di triple sono utilizzati come input per la ricerca semantica. Dalla ricerca semantica deriva un insieme di risultati rispetto ad ogni query SPARQL.

Per produrre le metriche di confronto, si tiene in considerazione la rilevanza dei risultati. Un risultato è definito rilevante se è presente nell'insieme di risultati esatti della ground truth. Quindi, la rilevanza sarà la base dei calcoli quantitativi successivi da cui saranno trattate le conclusioni finali.

3.2 Ground truth

La *ground truth* è composta da un insieme di query, ciascuna delle quali è associata a un insieme di risultati esatti. Pertanto, per costruire la ground truth, è necessario generare le query su cui verrà effettuata la valutazione e ricavare i risultati esatti corrispondenti per ogni query. La generazione delle query si basa su alcuni concetti derivanti dal calcolo combinatorio.

3.2.1 Generazione delle query

Le query della ground truth sono costruite tramite la combinazione di caratteristiche definite all'interno del corpus dei dati.

Al fine di generare query non affette da bias, la generazione delle stesse si basa sulla selezione di N caratteristiche da un insieme di M caratteristiche totali, e le combinazioni possibili sono date dal *coefficiente binomiale*. Questo coefficiente, denotato con $\binom{M}{N}$, rappresenta il numero di modi in cui è possibile scegliere N elementi distinti da un insieme di M elementi, ed è definito dalla seguente formula:

$$\binom{M}{N} = \frac{M!}{N!(M-N)!} \tag{3.1}$$

dove $M!$ è il fattoriale di M.

Per generare tutte le possibili query utilizzando tutte le M caratteristiche, è necessario considerare non solo le combinazioni di un singolo numero N, ma tutti i valori che possono essere assunti da N, ovvero tutti gli interi compresi tra 1 e M. Il numero totale di combinazioni possibili è dato dalla somma dei coefficienti binomiali per tutti i valori di N da 1 a M, e si può calcolare come:

$$\sum_{N=1}^{M} \binom{M}{N} = 2^M - 1 \tag{3.2}$$

Questo valore rappresenta tutte le possibili combinazioni di feature, escluse l'insieme vuoto. Queste combinazioni costituiranno l'insieme di query di base utilizzate per la generazione della ground truth.

Per esempio, nel caso specifico del nostro caso di studio, sono utilizzati i valori di $N = 2, 3, 4$ e $M = 5$. La scelta di questi valori è causata dalla necessità di generare un numero di query sufficientemente grande su cui poter calcolare i risultati e le relative metriche, ma che sia anche eseguibile in tempo ragionevole dalla macchina su cui viene eseguito lo studio. Utilizzando la formula del coefficiente binomiale, possiamo calcolare il numero di combinazioni per ciascun N:

$$\binom{5}{2} = \frac{5!}{2!(5-2)!} = 10, \quad \binom{5}{3} = \frac{5!}{3!(5-3)!} = 10, \quad \binom{5}{4} = \frac{5!}{4!(5-4)!} = 5$$

Pertanto, per $N = 2, 3, 4$, abbiamo, rispettivamente, 10, 10 e 5 combinazioni di caratteristiche.

Ogni combinazione di caratteristiche può generare un numero variabile di query, a seconda di quanti valori ciascuna caratteristica può assumere. Il numero di valori possibili per ogni caratteristica dipende quindi dal suo dominio, e questo influisce direttamente sul numero totale di query che è necessario generare per ottenere la ground truth. Gli aspetti implementativi della ground truth applicati ad un sito di eCommerce sono trattati nel Capitolo 4.

3.2.2 Risultati esatti

Per ogni query di base generata mediante il calcolo combinatorio, si procede ad estrarre i risultati associati utilizzando le informazioni presenti nel corpus dei dati. Un elemento sarà inserito nell'insieme dei risultati esatti se rispetta i valori per le caratteristiche richieste dalla query.

Questo processo consiste nell'applicare una serie di filtri all'intero insieme di elementi presenti nel corpus dei dati, in modo che ogni elemento nei risultati soddisfi i vincoli imposti dai valori delle caratteristiche specificate nella query. L'implementazione dei filtri dipende dalla rappresentazione specifica del corpus dei dati. Per esempio, nel caso di studio, per ogni query composta da caratteristiche con valori specifici, il codice ricopia tutti gli elementi del corpus nell'insieme dei risultati e rimuove gli elementi che non rispettano i vincoli sui valori imposti dalla query.

L'applicazione di questi filtri garantisce che ogni elemento nei risultati soddisfi i vincoli imposti dai valori delle caratteristiche specificate nella query. I risultati ottenuti in questo modo rappresentano una ground truth accurata, in quanto provengono direttamente dal corpus e sono quindi considerati esatti. Inoltre, questo metodo di filtraggio è neutrale rispetto alle strategie di ricerca, poiché si basa esclusivamente sui dati disponibili e sui criteri definiti nelle query.

Per ridurre la quantità di query da considerare, si selezionano solo quelle query che restituiscono un numero minimo di risultati. Per esempio, nel caso di studio il numero minimo di risultati è fissato a 20. A questo punto, la ground truth è completa e contiene sia le query di base, impiegate per la generazione di query testuali e in SPARQL, sia i relativi risultati esatti, che vengono utilizzati per la valutazione delle prestazioni dei due approcci.

3.3 Strategie di ricerca

Le strategie di ricerca oggetto di confronto sono la ricerca basata su parole chiave e la ricerca semantica. Questi due approcci rappresentano soluzioni diverse per affrontare il problema del ritrovamento di informazioni rilevanti all'interno di grandi insiemi di dati.

Nel nostro contesto di utilizzo, i due metodi di ricerca restituiscono un insieme di risultati identificati come rilevanti rispetto ad una query in input, con la rilevanza espressa in modo binario, secondo cui un documento è considerato o rilevante o non rilevante alla query. Un risultato è rilevante se è presente nel ground truth. L'alternativa alla rilevanza binaria è la rilevanza graduata, che usa una scala discreta per distinguere il grado di rilevanza dei documenti, consentendo anche un ordinamento dei risultati. Questo tipo di rilevanza non è stato usato nel framework proposto in questo lavoro di tesi, ma è lasciato come direzione per possibili sviluppi futuri.

La ricerca basata su parole chiave è uno dei metodi più tradizionali e diffusi nei motori di ricerca. L'algoritmo di ricerca tenta di trovare una corrispondenza tra tutte le parole inserite dall'utente nella query e quelle contenute nei documenti presenti nel corpus di dati. Sebbene sia un approccio di semplice implementazione, è limitato dal fatto che nella sua forma base non considera il significato o il contesto dei termini di ricerca utilizzati nella query. La corrispondenza si basa infatti solo sulla presenza o assenza di termini presenti nella query.

A seconda delle implementazioni di questo approccio di ricerca, si potrebbero affrontare problemi come l'omonimia (quando una parola ha più significati), la sinonimia (quando diversi termini hanno lo stesso significato) o la ricerca multilingue (quando i termini appaiono in lingue diverse). Questi aspetti vengono spesso gestiti tramite sistemi di similarità semantica, che cercano di interpretare il significato delle parole all'interno del contesto. Tuttavia, nel framework sviluppato in questa tesi ci si limiterà a operazioni più semplici come lo stemming, che riconduce le parole alla loro forma base per facilitare la corrispondenza.

Con l'avvento del Web semantico si è diffusa una nuova strategia di ricerca, la ricerca semantica, che sfrutta le annotazioni prodotte per descrivere le risorse. Questo approccio si distingue dalla ricerca per parole chiave perché è scritta in un linguaggio strutturato e comprensibile alle macchine, le quali possono quindi comprendere il significato semantico dei termini di ricerca e restituire risultati potenzialmente più precisi e completi. La qualità e la quantità di informazioni associate ad ogni elemento nel catalogo influenzano direttamente le prestazioni della ricerca semantica.

3.3.1 Ricerca basata su parole chiave

La ricerca basata su parole chiave, o basata su keywords, restituisce come elementi rilevanti tutti gli elementi del catalogo che presentano una corrispondenza con tutti i termini di ricerca all'interno della query testuale.

Nel contesto del framework proposto, è necessario ottenere le query testuali da utilizzare in questo approccio di ricerca. Se le query di base interne alla ground truth sono generate mediante una combinazione di caratteristiche e la loro successiva attribuzione di valori, le query testuali possono essere costruite cercando esplicitamente tali valori come termini di ricerca, o parole chiave. Il processo, dal punto di vista implementativo, è analizzato nel Capitolo 4.

La ricerca basata su parole chiave svolge come prima operazione la *tokenizzazione*, che consiste nel suddividere le stringhe di testo in unità più piccole, i token. In questo caso i le stringhe di testo sono le query testuali e i token sono le parole, che rappresentano i termini significativi e saranno analizzati nel processo di ricerca.

Prima del confronto con la descrizione di ogni elemento, i token vengono sottoposti ad una operazione di *stemming*. Lo stemming è una tecnica che riduce le parole alle loro radici, eliminando affissi per trattare parole diverse come varianti di una singola radice. Ad esempio, con la query testuale "fragranze uomo agrumato", la parola "agrumato", che si riferisce alla categoria olfattiva del prodotto, sarà ridotta alla sua radice "agrum" per trovare corrispondenze anche con parole del tipo "agrumate", "agrumati", "agrumi" e tutte le altre variazioni possibili. In questo modo forme derivate, plurali e forme coniugate delle parole all'interno della query testuale vengono comunque considerate.

Una tecnica più avanzata per gestire le variazioni è la *lemmatizzazione*, ma non è utilizzata nella ricerca basata su parole chiave adottata in questa tesi. La lemmatizzazione consiste nel ridurre le parole alla loro forma base, tenendo conto del contesto grammaticale e semantico, permettendo così di riconoscere forme verbali e nominali diverse come equivalenti. Questo approccio è particolarmente utile in situazioni in cui il significato di una parola può variare a seconda del suo uso in una frase, migliorando così la capacità di recupero delle informazioni, ma richiede un'analisi più approfondita e risorse computazionali superiori rispetto allo stemming,

rendendolo meno praticabile in scenari di ricerca basati esclusivamente su parole chiave.

Dopo aver ottenuto i token e averli ridotti alla loro radice tramite lo stemming, la ricerca basata su parole chiave cerca una corrispondenza tra tutti i token che compongono la query e le descrizioni e i titoli degli elementi. I risultati sono dati da quegli elementi che presentano una corrispondenza nel titolo e/o nella descrizione di tutti i token.

Uno dei principali vantaggi della ricerca basata su parole chiave è la sua semplicità di implementazione, che la rende particolarmente adatta per la gestione di grandi volumi di dati. Infatti, oltre ad essere un sistema semplice ed efficace da implementare, può essere utilizzato in modo intuitivo anche da parte dell'utente, che deve scrivere una serie di parole chiave che descrivono ciò che desidera e non necessita di alcuna conoscenza pregressa di alcun linguaggio di interrogazione.

Lo svantaggio maggiore è che questo approccio non presenta una comprensione profonda del significato dei token e di conseguenza non è adatto a query più complesse. Per esempio, se svolgiamo una query del tipo "fragranze donna minori di 50 euro" su un sistema di ricerca basato su keywords, il sistema potrebbe non riuscire a trovare i prodotti della categoria "fragranze donna" che hanno un prezzo effettivamente minore di 50 Euro, in quanto non ha alcuna comprensione del significato di ogni token, a meno che nel prodotto non sia esplicitamente scritto "minore di 50 euro". I token vengono infatti sono trattati tutti nello stesso modo, cercando solamente una corrispondenza, nella descrizione o nel titolo del prodotto.

3.3.2 Ricerca semantica

La ricerca semantica, a differenza della ricerca basata su parole chiave, adotta un approccio che sfrutta le annotazioni semantiche per migliorare la corrispondenza tra query e dati. Vengono usate le triple RDF per rappresentare le informazioni in una forma composta da soggetto, oggetto e un predicato che definisce il tipo di relazione tra il soggetto e l'oggetto. In questo modo, il significato semantico di ogni valore viene esplicitato ed è reso interpretabile dalle macchine.

Le query utilizzate per la ricerca semantica sono invece espresse in linguaggio

SPARQL, linguaggio standard per interrogare basi di dati semantiche che modellano i dati in triple RDF. Nelle query SPARQL, l'utente definisce esattamente le proprietà che gli elementi ricercati devono possedere. Ogni clausola della query specifica relazioni e attributi che devono essere soddisfatti, espressi attraverso *triple pattern*, o pattern di triple, che rappresentano la struttura di base per il recupero dei dati semantici. Un pattern di triple è una combinazione di soggetto, predicato e oggetto, in cui alcuni o tutti i componenti possono essere variabili. Questi pattern vengono utilizzati per confrontare le triple nel grafo RDF, alla ricerca di quelle che soddisfano le condizioni definite nella query.

Ad esempio, una query SPARQL per un sito di eCommerce potrebbe specificare non solo che un prodotto deve appartenere a una certa categoria, ma anche che deve avere determinati attributi e soddisfare vincoli numerici, come il prezzo o la capacità. In pratica, si cercano quegli elementi descritti da triple che hanno determinati valori per determinate caratteristiche.

Per poter ricavare le query SPARQL a partire dalle query nella ground truth, ci sono diversi approcci. Un primo approccio potrebbe prevedere l'uso di modelli di linguaggio di grandi dimensioni (Large Language Models, LLM) per generare automaticamente le query SPARQL, prendendo in input le query di base e seguendo le istruzioni di un prompt ingegnerizzato appositamente per l'operazione. Tuttavia, questo approccio è stocastico, il che significa che non potrebbe essere sempre preciso e potrebbe produrre risultati scorretti. In alternativa, come viene fatto nel caso di studio, si costruisce la query SPARQL in modo automatico in base alle caratteristiche presenti nella query di base. La presenza di una caratteristica corrisponde a una clausola da avvalorare nel corpo della query SPARQL. In questo modo, se le clausole sono formulate correttamente e definiscono i pattern di triple in base alla struttura dei dati, si ottengono query SPARQL sempre valide. Il processo è descritto dal punto di vista implementativo nel Capitolo 4.

Un vantaggio significativo della ricerca semantica è la possibilità di utilizzare direttamente le annotazioni semantiche, in modo da ottenere risultati estremamente mirati, poiché le query SPARQL possono fare riferimento a concetti più ricchi e articolati rispetto a quanto possibile con una semplice corrispondenza testuale. Un aspetto avanzato è che la ricerca semantica è in grado di eseguire inferenze o

ragionamenti complessi, se svolta sotto un regime di *entailment* (inferenza logica), che consente di dedurre nuovi fatti a partire dalle regole esistenti.

La qualità dei risultati della ricerca semantica dipende fortemente dalla completezza e dall'accuratezza delle informazioni presenti nel grafo di triple RDF su cui viene eseguita. Le informazioni sugli elementi, infatti, devono essere sufficientemente ricche e strutturate, e devono riferirsi a vocabolari condivisi per garantire che il significato semantico sia interpretato correttamente. Naturalmente, in mancanza di un'adeguata annotazione delle risorse, i risultati potrebbero esserne inficiati.

3.4 Metriche di valutazione

Le query testuali e le query SPARQL vengono eseguite sullo stesso insieme di elementi, sebbene rappresentati in formati diversi, al fine di ottenere un insieme di risultati derivante dall'approccio basato su corrispondenza di parole chiave e un insieme di risultati derivante dall'esecuzione di query in SPARQL. Questi insiemi sono relativi alla stessa query di base e, quindi, sono direttamente confrontabili per il calcolo delle prestazioni. Per valutare la qualità dei risultati ottenuti, gli insiemi dei risultati vengono confrontati con l'insieme di risultati esatti della ground truth, e da tale confronto si calcolano diverse metriche quantitative.

Le metriche di valutazione sono fondamentali per analizzare e confrontare in maniera numerica e oggettiva l'efficacia dei due sistemi di ricerca. In particolare, l'analisi si concentra su due metriche principali, la *precisione* e il *richiamo*, e ad una metrica che le combina, la *F1 measure*.

3.4.1 Precisione

La *precisione*, in inglese *precision*, è definita come la proporzione di risultati rilevanti tra tutti i risultati restituiti da un sistema di ricerca. Essa rappresenta un indicatore della qualità delle risposte fornite, misurando la capacità del sistema di restituire solo i risultati pertinenti in risposta a una query. Se consideriamo come risultati rilevanti i risultati presenti nella ground truth e come risultati restituiti quelli restituiti dal sistema oggetto di valutazione, la formula per calcolare la precisione è:

$$\text{Precisione} = \frac{\text{Numero di risultati rilevanti restituiti}}{\text{Numero totale di risultati restituiti}} \tag{3.3}$$

Confrontando l'insieme dei risultati restituiti dal sistema con i risultati esatti associati a ciascuna query nella ground truth, è possibile definire la precisione in base alla corrispondenza tra gli elementi presenti nei due insiemi. Si definiscono quindi quattro classi di risultati, basate sulla distinzione tra "positivi" e "negativi". Un elemento è considerato "positivo" se è restituito come risultato, mentre è "negativo" se non è presente nell'insieme dei risultati. In base alla corrispondenza con l'insieme dei risultati esatti, distinguiamo come "veri" i risultati che sono riportati nello stesso modo sia nell'insieme da valutare che nell'insieme dei risultati esatti, mentre sono considerati "falsi" se i due risultati non combaciano. Le quattro classi di risultati sono quindi definite come segue:

- *Veri positivi* (TP): risultati rilevanti che sono stati correttamente restituiti dal sistema.
- *Falsi positivi* (FP): risultati non rilevanti che sono stati erroneamente restituiti dal sistema.
- *Falsi negativi* (FN): risultati rilevanti che non sono stati restituiti dal sistema.
- *Veri negativi* (TN): risultati non rilevanti che sono stati correttamente esclusi dal sistema.

Pertanto, la precisione può essere definita anche come il rapporto dei veri positivi (TP) sul totale dei positivi, ottenuto dalla somma tra veri positivi e falsi positivi ($TP + FP$). Quindi, si può esprimere la precisione anche nel modo seguente:

$$\text{Precisione} = \frac{TP}{TP + FP} \tag{3.4}$$

Se la precisione assume valori molto vicini a 1, significa che i risultati restituiti dal sistema sono molto corretti. Al contrario, se la precisione è vicina a 0, indica una proporzione elevata di risultati erronei nell'insieme totale.

3.4.2 Richiamo

Il *richiamo*, in inglese *recall*, misura la capacità del sistema di recuperare i risultati rilevanti tra tutti i risultati rilevanti disponibili per una determinata query. Il richiamo rappresenta un indicatore della completezza delle risposte fornite, mostrando quanto efficacemente il sistema riesca a identificare tutti i risultati pertinenti. Se consideriamo come risultati rilevanti quelli presenti nella ground truth e come risultati restituiti quelli forniti dal sistema oggetto di valutazione, la formula per calcolare il richiamo è:

$$\text{Richiamo} = \frac{\text{Numero di risultati rilevanti restituiti}}{\text{Numero totale di risultati rilevanti}} \tag{3.5}$$

Analogamente alla precisione, si può definire il richiamo utilizzando le stesse quattro classi di risultati già definite, basate sulla distinzione tra risultati positivi o negativi, veri o falsi. In particolare, il richiamo è definito come il rapporto tra i veri positivi (TP) e il totale dei risultati veri, ovvero la somma tra veri positivi e falsi negativi ($TP + FN$):

$$\text{Richiamo} = \frac{TP}{TP + FN} \tag{3.6}$$

Per valori molto vicini ad 1, il richiamo indica che il sistema è in grado di trovare quasi tutte le risposte corrette. Se invece il richiamo è molto vicino a 0, indica al contrario che il sistema non riesce a trovare un numero significativo di risposte corrette sul totale.

3.4.3 F1 Measure

La *F1 measure* è una metrica che combina precisione e richiamo in un unico valore, fornendo un equilibrio tra i due. Questa misura è particolarmente utile quando si desidera avere un'idea complessiva delle prestazioni di un sistema, specialmente in presenza di uno squilibrio tra precisione e richiamo. La F1 measure è definita come la media armonica tra precisione e richiamo, ed è calcolata con la seguente formula:

$$F_1 = 2 \cdot \frac{\text{Precisione} \cdot \text{Richiamo}}{\text{Precisione} + \text{Richiamo}}. \tag{3.7}$$

Alternativamente, può essere espressa in termini di veri positivi (TP), falsi positivi (FP), e falsi negativi (FN) come segue:

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{3.8}$$

Questa formula consente di ottenere un valore compreso tra 0 e 1, dove un valore più vicino a 1 indica prestazioni migliori del sistema di ricerca. La F1 measure è particolarmente utile quando si desidera massimizzare sia la precisione che il richiamo simultaneamente, rendendola una metrica di riferimento nelle valutazioni dei sistemi di informazione.

In ogni caso, per avere una panoramica completa di un sistema di ritrovamento, è stato deciso di utilizzare tutte e tre le metriche quantitative, poiché forniscono informazioni complementari per valutare l'efficacia del sistema. L'analisi combinata di precisione, richiamo e F1 measure consente di ottenere un quadro più dettagliato delle prestazioni del sistema, per comprendere come un sistema si comporta meglio dell'altro sotto determinate condizioni.

Capitolo 4

Applicazione del Framework ad un sito di eCommerce

In questo capitolo viene applicato il framework proposto nel Capitolo 3 ad un sito Web di eCommerce. Viene introdotto il sito Web di eCommerce fornito dall'azienda Sidea Group S.r.l. e vengono applicate dal punto di vista implementativo le fasi che costituiscono il framework. Si applica la fase di *annotazione*, mostrando le tecnologie applicate nella costruzione delle annotazioni semantiche. Quindi, si descrive la costruzione del *corpus dei dati* a partire dal database del sito. Si prosegue con il *raffinamento* delle annotazioni, mostrando l'utilizzo di strumenti di convalida del markup e applicando modifiche al grafo di triple RDF che descrivono i prodotti.

Successivamente, si applica la fase di *valutazione*. Per questa fase si parte con la creazione della *ground truth* a partire dal corpus dei dati costruito precedentemente. Quindi, si procede con la descrizione della procedura impiegata per la generazione di query in linguaggio naturale e query in linguaggio SPARQL a partire dalla ground truth. Inoltre, viene presentato un metodo di conversione da query testuali a query SPARQL, con l'ausilio di *Large Language Model* (LLM) per automatizzare l'interpretazione delle query. Alla fine del capitolo, sono mostrati i dettagli implementativi sul calcolo delle metriche per la valutazione quantitativa.

Il caso di studio analizza il ruolo che svolge l'annotazione semantica in un sito Web di eCommerce, attualmente in utilizzo. In particolare, si vuole mostrare come

i dati strutturati associati ai prodotti possono essere impiegati direttamente per la ricerca di prodotti. L'obiettivo è dimostrare quantitativamente che gli strumenti di ricerca semantica siano in grado di rispondere a query molto specifiche e complesse.

Per misurare il miglioramento derivante dall'utilizzo di dati annotati semanticamente nella ricerca di un prodotto, si confronta la ricerca semantica con la ricerca attualmente impiegata all'interno del sito basata sulla corrispondenza tra le parole chiave della query inserita dall'utente e le parole presenti nella descrizione e nel titolo dei prodotti (come descritto nel Capitolo 3).

Per poter arrivare ad una misura quantitativa delle prestazioni dei due sistemi, si è applicato il framework ad un caso reale oggetto del business aziendale. Infatti, si è partiti dal sito Web di eCommerce reso disponibile dall'azienda, verificando la correttezza delle annotazioni semantiche esistenti in formato JSON-LD ed integrandole di dati mancanti. Si è quindi passati alla costruzione di un *corpus dei dati* con tutte le informazioni principali sui prodotti, prelevandole dal database collegato al sito Web. Dal corpus dei dati è stata generata una ground truth, ossia, un insieme di query associate ai rispettivi risultati noti a priori. Le query in tale insieme sono generate come combinazione degli elementi di un insieme predefinito di caratteristiche dei prodotti provenienti dal corpus dei dati. Ottenute le equivalenti query testuali e in SPARQL, sono state eseguite sul catalogo e i relativi risultati sono confrontati con la ground truth per la misura quantitativa delle prestazioni.

Migliorando la correttezza dei risultati presentati all'utente, si può potenzialmente incrementare il numero di vendite e di visite al sito. È pertanto di fondamentale importanza per un sito di eCommerce assicurarsi che vengano restituiti al cliente i risultati più attenenti alla propria interrogazione.

È possibile consultare il repository di GitHub per il presente progetto di tesi al seguente link: `https://github.com/frankcesco/Tesi_Reggio_2024`.

4.1 Sito Web di eCommerce

Il sito di eCommerce reso disponibile dall'azienda è il sito Web di Ethos Profumerie, dedicato alla vendita al dettaglio di profumi e prodotti per il benessere della persona. La piattaforma è basata su WordPress[1], con plugin per la gestione dei prodotti e delle funzionalità di vendita. Ogni prodotto presente nel catalogo è gestito come un singolo *post* di WordPress, e quindi le informazioni descrittive sui prodotti fanno riferimento direttamente al post corrispondente del prodotto.

Per il caso di studio, è stata utilizzata una versione locale del sito e sono stati gestiti un totale di 6420 prodotti. Non è stata utilizzata la versione del sito in produzione per poter applicare le modifiche senza compromettere l'esercizio dell'attività.

L'intero sistema software del sito Web è stato organizzato all'interno di un ambiente Docker, un software open-source che permette di eseguire applicazioni in ambienti isolati, in modo tale da facilitare la gestione delle dipendenze, garantire la portabilità del sistema su diverse piattaforme e semplificare la distribuzione e l'integrazione dei vari componenti software. L'ambiente del sito è suddiviso in container distinti, ciascuno con una funzione specifica. Ad esempio, un container è dedicato al server Web NGINX, un altro alla piattaforma WordPress, mentre altri container gestiscono il database MariaDB e l'interfaccia phpMyAdmin. Ogni container può essere avviato, fermato o modificato indipendentemente, ma per l'esecuzione completa del sito è necessario che tutti i container siano avviati e attivi contemporaneamente.

È inoltre presente un container dedicato che consente di interagire con il sistema tramite la *Command Line Interface* (CLI) per eseguire comandi di manutenzione e gestione dei contenuti. Utilizzando la CLI, è stato sviluppato un plugin di WordPress che accetta come argomento la query testuale e la esegue direttamente sul catalogo del sito, esportando i risultati come liste di ID serializzate su disco. In questo modo, è stato possibile automatizzare l'esecuzione delle query testuali attraverso il plugin del sito che si occupa di eseguire la ricerca basata su parole chiave,

[1]WordPress è un Content Management System per il Web: `https://wordpress.com`

Clerk.io. Il plugin è utilizzato nella fase di valutazione del framework, per produrre i risultati della ricerca basata su parole chiave.

La Figura 4.1 mostra uno screenshot della versione in locale del sito. La schermata mostra la sezione del sito dedicata ai profumi disponibili, in ordine alfabetico. Da notare come sulla sinistra, i profumi siano catalogati secondo gruppo e famiglia olfattiva, oltre che per il brand.

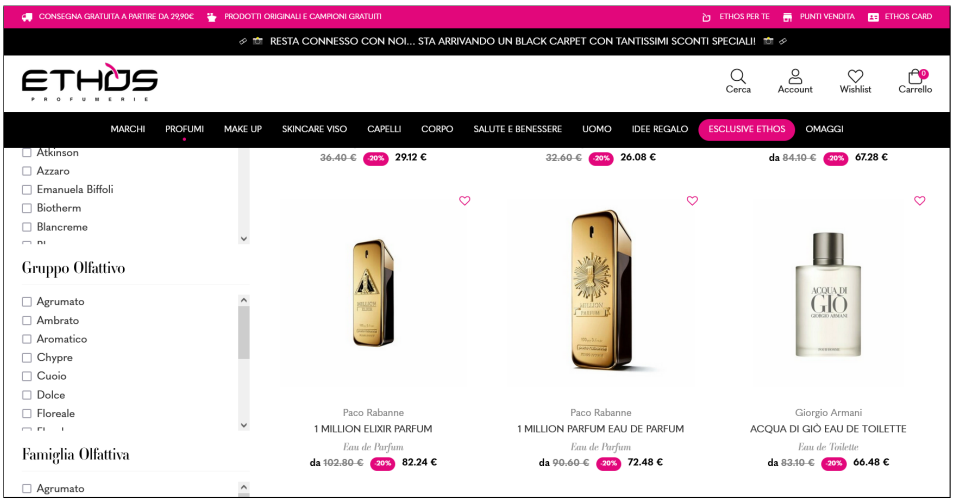

Figura 4.1: Screenshot del sito Web oggetto del caso di studio, nella sezione dedicata ai profumi.

4.2 Dati dei prodotti

Seguendo il framework introdotto nel Capitolo 3, è necessario partire con la fase di *annotazione*, in cui si applica l'annotazione semantica alle pagine Web del sito di eCommerce oggetto di studio. Innanzitutto, va esaminato il sito Web per verificare se ci sono annotazioni esistenti ed in caso contrario bisogna valutare il formato di annotazione da adottare, in base alla modalità di generazione delle pagine Web [12].

4.2.1 Annotazione semantica dei prodotti

Nel caso del sito Web oggetto di studio, iI prodotti nelle pagine Web erano già annotati semanticamente nel formato JSON-LD e fanno riferimento al vocabolario condiviso Schema.org. Le annotazioni semantiche sono generate automaticamente

dal plugin di WordPress *Rank Math SEO*[2], il quale si occupa delle operazioni di Search Engine Optimization (SEO). Questo plugin compila le informazioni strutturate prelevandole direttamente dai metadati di WordPress o da informazioni salvate in altri plugin, in particolare da quelli che abilitano il sito per l'eCommerce, come il plugin *WooCommerce*[3].

Per ottenere un unico file che includa tutte le triple RDF dei prodotti in un grafo, su cui poter eseguire query in linguaggio SPARQL, è stato necessario prelevare i dati strutturati dei prodotti dalle pagine e unificarli in un singolo grafo. Siccome i dati sono generati automaticamente dal plugin, è stato necessario prima generare la pagina HTML e poi prelevarli direttamente dal tag `<script>`. Per fare ciò, è stato sviluppato uno script in linguaggio Python che effettua uno *scraping* delle pagine dei prodotti, ossia estrae da esse le informazioni necessarie e le esporta in un unico file JSON.

Nel caso di studio, lo script legge una lista di 6240 URL del sito in locale e invia una richiesta HTTP a ciascuna pagina del prodotto. Per ogni pagina, scarica il contenuto HTML e utilizza una libreria di parsing per analizzarlo. All'interno delle pagine HTML, lo script cerca i tag `<script>` contenenti dati in formato JSON-LD relativi ai prodotti, identificati dal campo `"@type": "Product"`. Successivamente, viene creato un file JSON in cui ogni prodotto è aggiunto mediante un'operazione di `append` sotto il campo `"@graph"`. Ogni prodotto è identificato per mezzo di URI, sotto il campo `"@id"`, che corrisponde all'URL della pagina. Questo processo include anche la gestione degli errori, per garantire che il programma possa continuare a funzionare anche in caso di problemi con alcune pagine.

Alla fine del processo, si avrà un unico file JSON, che potrà essere convertito in altri formati di serializzazione di triple RDF come Turtle, per facilitare le interrogazioni successive. In Figura 4.2 è mostrato un esempio di prodotto esportato dallo script di scraping, senza modifiche successive. Ogni prodotto come quello in figura è inserito in un grafo, identificato da `"@graph"` e sotto il contesto `"@context": "https://schema.org"`. Su questo grafo si potranno eseguire le query SPARQL.

[2] `https://rankmath.com`

[3] `https://woocommerce.com/woocommerce`

```
{
  "@type": "Product",
  "brand": {
    "@type": "Brand",
    "name": "Ck One"
  },
  "name": "Calvin Klein | CK ONE | Deodorant Stick 75ml",
  "description": "CK One Deodorante Stick ti stimola a sentirti audace e pulito.
  ↪  Note di rinfrescante tè verde si unisce a rosa e viola, culminando in un
  ↪  finale di muschio ed ambra.",
  "sku": "03760_60705",
  "category": "Profumi",
  "mainEntityOfPage": {
    "@id": "http://www.ethos.local/prodotto/deodorant-stick-75ml/#webpage"
  },
  "offers": {
    "@type": "Offer",
    "price": "18.75",
    "priceCurrency": "EUR",
    "priceValidUntil": "2025-12-31",
    "availability": "https://schema.org/InStock",
    "itemCondition": "NewCondition",
    "url": "http://www.ethos.local/prodotto/deodorant-stick-75ml/",
    "seller": {
      "@type": "Organization",
      "@id": "http://www.ethos.local/",
      "url": "http://www.ethos.local"
    },
    "priceSpecification": {
      "price": "18.75",
      "priceCurrency": "EUR",
      "valueAddedTaxIncluded": "true"
    }
  },
  "@id": "http://www.ethos.local/prodotto/deodorant-stick-75ml/#richSnippet",
  "image": {
    "@id": "http://www.ethos.local/wp-content/uploads/2009/10/
    ↪  27797-Calvin-Klein-CK-ONE-Deodorant-Stick-75ml.jpg"
  }
}
```


Figura 4.2: Esempio di dati strutturati in formato JSON-LD per un prodotto del caso di studio.

4.2.2 Creazione del corpus dei dati

Prima di passare alla fase di *raffinamento*, è stato necessario creare il corpus dei dati, che include informazioni aggiuntive sui prodotti che verranno integrate nei

dati strutturati. La creazione del *corpus dei dati* è inoltre il primo passo verso la costruzione della ground truth per la valutazione delle prestazioni.

A partire dalla versione del sito salvata sulla macchina locale, `ethos.local`, è stata utilizzata l'interfaccia di amministrazione del database phpMyAdmin per eseguire una query SQL, riportata in Figura 4.3. La query SQL è stata progettata per estrarre una lista distinta di prodotti pubblicati dal database, includendo per ciascun prodotto un *permalink* completo, generato concatenando l'URL base del sito con il valore del campo `post_name`. Oltre al *permalink*, la query restituisce l'ID univoco del prodotto, il titolo (`post_title`) e la descrizione completa (`post_content`). Tramite una sotto-query, è stato possibile recuperare le categorie associate a ciascun prodotto, le quali sono aggregate in una singola stringa, separate da virgole, utilizzando la funzione `GROUP_CONCAT`. La query ha filtrato i risultati in modo che venissero considerati solo i prodotti con il campo `post_name` non nullo e non vuoto, e che avessero lo stato `'publish'`, assicurando così che solo i prodotti attualmente pubblicati fossero inclusi nel risultato.

La query SQL ha restituito una tabella di 6240 prodotti, ciascuno dei quali era già associato nel database alla sua categoria, al titolo e alla descrizione. Ogni prodotto è stato identificato tramite il suo URL e un ID sequenziale interno al sito.

```sql
SELECT DISTINCT
       CONCAT('http://www.ethos.local/prodotto/', p.post_name) AS permalink,
       p.ID,
       p.post_title,
       p.post_content,
       -- Sotto-query per ottenere la categoria
       (SELECT GROUP_CONCAT(DISTINCT c.name ORDER BY c.name ASC SEPARATOR ', ')
        FROM wbp_term_relationships tr
        JOIN wbp_term_taxonomy tt ON tr.term_taxonomy_id = tt.term_taxonomy_id
        JOIN wbp_terms c ON tt.term_id = c.term_id
        WHERE tr.object_id = p.ID
          AND tt.taxonomy = 'product_cat') AS Categorie
FROM `wbp_posts` p
WHERE p.post_type = 'product'
  AND p.post_name IS NOT NULL
  AND p.post_name <> ''
  AND p.post_status = 'publish';
```

Figura 4.3: Query SQL per ottenere permalink, ID, nome, descrizione e categorie di prodotto dal database del sito di eCommerce.

Le informazioni ottenute dalla query SQL in Figura 4.3 sono utilizzate per estrarre una serie di dati aggiuntivi, ovvero le caratteristiche da impiegare nella costruzione delle query della ground truth attraverso il calcolo combinatorio.

Per ottenere queste caratteristiche aggiuntive sui prodotti, è stato necessario ricavarle dal titolo e dalla descrizione dei prodotti. In particolare, siccome il dominio è composto da profumi e oggetti per la cura della persona, è stato posto l'obiettivo di ricavare come caratteristiche aggiuntive dei prodotti le *categorie olfattive* e le *capacità*, in millilitri. In questo modo, le query nella ground truth possono far riferimento anche a queste caratteristiche sui prodotti. Le capacità sono inserite all'interno dei titoli dei prodotti, quindi si estraggono tramite matching di espressioni regolari. Per le categorie olfattive, invece, il processo richiede l'utilizzo di un LLM.

4.2.3 Classificazione con Large Language Model

Per poter ricavare le categorie olfattive, viene adottato un LLM per estrarre dalle descrizioni testuali dei prodotti le categorie olfattive a cui appartengono, a partire da un insieme predefinito di tali categorie. Questo passaggio contribuisce a completare il corpus dei dati a partire dal quale verrà creata la ground truth. L'insieme delle categorie olfattive deriva dai filtri della ricerca interna al sito, come rappresentato in Figura 4.1. Sebbene queste informazioni fossero presenti come filtri di ricerca, non erano codificate all'interno del database del sito, pertanto è stato necessario ricavarle dalle descrizioni dei prodotti.

Un LLM è un sistema di intelligenza artificiale progettato per processare e generare testo in modo coerente, con capacità di generalizzazione su una vasta gamma di compiti linguistici. I LLM sono caratterizzati da una significativa quantità di parametri, generalmente nell'ordine di decine o centinaia di miliardi, e sono addestrati su grandi quantità di dati testuali, tipicamente nell'ordine di gigabyte o terabyte. Questi modelli sfruttano approcci di apprendimento auto-supervisionato e possono essere raffinati mediante il fine-tuning per compiti specifici, il che consente loro di eseguire compiti complessi come traduzione, sintesi e conversazione [20].

Il modello scelto per il task di classificazione è *Llama* 3 di Meta, specificamente

la versione con 70 miliardi di parametri, con il fine-tuning svolto da Groq[4]. Grazie all'uso di tecniche avanzate di fine-tuning, tra cui l'ottimizzazione della preferenza diretta (Direct Preference Optimization[5], DPO), questo modello open-source, sviluppato in collaborazione con Glaive[6], offre prestazioni superiori per l'uso di strumenti e le chiamate a funzioni [19].

In particolare, il modello `Llama-3-Groq-70B-Tool-Use`, usato per il caso di studio, ha raggiunto un'accuratezza complessiva del 90.76% sui benchmark, posizionandosi al primo posto nella *Berkeley Function Calling Leaderboard* (BFCL) al momento della pubblicazione del modello. Questo risultato è stato ottenuto grazie ad un'accurata analisi dei task, che includeva l'uso di funzioni e interazioni con API, dimostrando le capacità avanzate di questo modello nel trattare compiti complessi [19].

Uno dei parametri da impostare per utilizzare il modello è la `temperature`. La `temperature` è un *iperparametro* che controlla la casualità delle risposte generate da un modello di linguaggio. Un valore di `temperature` più basso (vicino a 0) tende a produrre risposte meno casuali e più prevedibili, in quanto il modello si attiene maggiormente ai dati di addestramento. Al contrario, un valore più alto (vicino a 1) promuove una maggiore varietà e creatività nelle risposte, ma potrebbe generare risposte inaffidabili. La `temperature` è stata impostata a 0.5 per favorire un equilibrio ottimale tra la varietà delle risposte generate e l'affidabilità delle informazioni fornite dal modello.

Nel codice Python sviluppato per la classificazione delle categorie olfattive, è stato inizialmente definito un *prompt*, mostrato nel codice in Figura 4.4. Un *prompt* è una sequenza di testo che viene fornita al LLM per guidarne il comportamento e influenzarne le risposte. Nel caso di studio, il prompt include una *variabile* di input, il *contesto*, le *istruzioni* e la descrizione dell'*output*.

---

[4]Groq è una società americana specializzata nello sviluppo di circuiti integrati specifici per applicazioni nel campo dell'intelligenza artificiale, nota per l'accelerazione delle prestazioni di inferenza dei LLM come *Llama*. `https://groq.com`

[5]Ulteriori informazioni sulla DPO possono essere trovate qui: `https://arxiv.org/abs/2305.18290`

[6]`https://glaive.ai`

```
system='''
<variabile>
{descrizione}
</variabile>

<contesto>
Sei un frontend developer e stai lavorando a un e-commerce. Il tuo compito è
↪  classificare i prodotti in modo dicotomico (0-1). La variabile {descrizione}
↪  contiene la descrizione di un prodotto. Classifica i prodotti in base alla
↪  loro descrizione nelle seguenti categorie:
1) Agrumato
2) Ambrato
3) Aromatico
4) Chypre
5) Cuoio
6) Dolce
7) Floreale
8) Fruttato
7) Gourmand
8) Legnoso
9) Muschiato
10) Senza Profumo
11) Speziato Leggero
</contesto>

<istruzioni>
1. Leggi la {descrizione} del prodotto.
2. Per ogni categoria assegna un punteggio 0 oppure 1 in base alla presenza o
↪  assenza di elementi caratteristici.
3. Esegui l'output in formato list di python, come indicato nel tag <output>.
</istruzioni>

<output>
[ ["Agrumato", punteggio], ["Ambrato", punteggio], ["Aromatico", punteggio],
↪  ["Chypre", punteggio], ["Cuoio", punteggio], ["Dolce", punteggio],
↪  ["Floreale", punteggio], ["Fruttato", punteggio], ["Gourmand", punteggio],
↪  ["Legnoso", punteggio], ["Muschiato", punteggio], ["Senza Profumo",
↪  punteggio], ["Speziato Leggero", punteggio] ]
</output>
'''
```

Figura 4.4: Prompt utilizzato con il LLM per la classificazione delle categorie olfattive, inserito nella variabile `system`.

Come mostrato in Figura 4.4, la `<variabile>` di input è indicata come la descrizione del prodotto, mentre il `<contesto>` aiuta il LLM a comprendere il ruolo che deve assumere e il compito che deve svolgere, fornendo informazioni utili sulle

aspettative e le modalità di classificazione. Le `<istruzioni>` indicano al modello i passi su come classificare i prodotti in categorie olfattive specifiche, assegnando un punteggio di 0 o 1 per ciascuna categoria, a seconda della presenza o meno di elementi caratteristici nella descrizione. In `<output>` viene invece specificato il formato dell'output che deve produrre il LLM, in questo caso una lista di Python.

Viene utilizzato *LangChain*[7], un framework che consente di rendere i prompt dinamici, adattandoli automaticamente a input diversi e facilitando la creazione di catene di azioni per eseguire operazioni sequenziali o parallele. In questo caso, LangChain è impiegato per adattare il prompt alle descrizioni dei prodotti e gestirne l'esecuzione programmatica su grandi quantità di dati. Il prompt utilizzato da Llama 3 è costruito con la classe `ChatPromptTemplate`, in cui l'input include la `descrizione` del prodotto. Il modello, configurato per operare in modalità di classificazione, riceve la descrizione del prodotto come input e restituisce un output formattato in una lista Python che contiene le categorie olfattive con i relativi punteggi.

Il ciclo principale dello script si occupa di iterare su ogni prodotto della tabella caricata in memoria, recuperando la sua descrizione e tentando di ottenere una classificazione delle categorie olfattive attraverso l'invocazione del modello `Llama-3-Groq-70B-Tool-Use`. Se l'invocazione ha successo, il risultato viene aggiunto a una lista di categorizzazione. Nel caso in cui si verifichino errori, come il superamento della lunghezza massima del contesto, il testo viene troncato e la richiesta viene ripetuta fino a un numero massimo di tentativi. Se invece non è possibile ottenere un risultato valido dopo il numero massimo di tentativi, viene aggiunto un valore di fallback predefinito che assegna punteggi nulli a tutte le categorie.

Eventuali errori di formattazione dovuti alla natura stocastica dei LLM vengono corretti con un altro script, che utilizza un nuovo prompt dedicato unicamente alla correzione di errori sintattici, con il modello `Llama-3-Groq-70B-Tool-Use` e il framework LangChain. Lo script itera unicamente sugli errori finché non vengono tutti completamente corretti.

[7] `https://www.langchain.com`

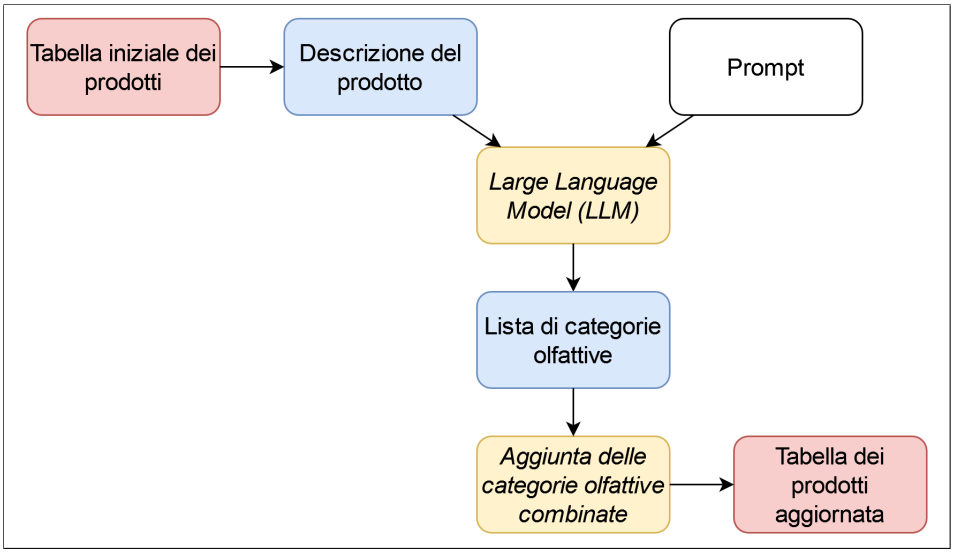

Figura 4.5: Schema che riassume la categorizzazione delle categorie olfattive.

La Figura 4.5 schematizza i passaggi principali della classificazione delle categorie olfattive. Dopo aver ottenuto le categorie olfattive di base, sono state calcolate con operazioni di concatenazione di stringhe tutte le categorie olfattive da due parole. Di queste categorie olfattive combinate, sono state prelevate le prime 10 aventi il maggior numero di prodotti associati a esse, in modo tale da ottenere in modo fedele sia i gruppi olfattivi singoli che le famiglie olfattive composte, come riportate nel sito. Alla fine del processo, è stata ottenuta una tabella dei prodotti aggiornata che include, insieme alle informazioni originarie, anche le informazioni sulle categorie olfattive, singole e doppie.

4.2.4 Raffinamento delle annotazioni

Per la fase di *raffinamento*, le pagine HTML dei prodotti sono state inizialmente sottoposte ai due strumenti di convalida del markup, esaminati nel Capitolo 3, che analizzano i dati strutturati interni ad ogni pagina e verificano che non ci siano errori critici, errori secondari ed informazioni mancanti.

Per quanto riguarda i risultati dello strumento di convalida di Schema.org, non sono stati segnalati errori o avvisi di alcun tipo. Questo significa che il codice è ben strutturato ed esente da errori. Lo strumento si è limitato a elencare quali dati strutturati sono stati inseriti nella pagina.

Lo strumento di Google, invece, ha rilevato che ci sono dei dati mancanti che si potrebbero aggiungere per fornire al cliente informazioni importanti sui prodotti. Questi dati mancanti riguardano le recensioni, con i campi `"schema:Review"` ed `"schema:aggregateRating"`. Altri dati segnalati come mancanti non si riferiscono ai prodotti ma alle schede commercianti, che potrebbero essere più complete con campi come `"schema:hasMerchantReturnPolicy"` e `"schema:shippingDetails"`. Questi campi forniscono dettagli su politiche di reso e di spedizione.

Questo vuol dire che, sebbene i dati siano strutturati correttamente all'interno delle pagine, il sito potrebbe beneficiare di dati aggiuntivi. Per il caso di studio, non sono state aggiunte le informazioni sulle recensioni, valuazioni, reso e spedizioni suggerite dallo strumento di Google perché andava oltre lo scopo di valutazione del progetto, ma si è deciso di completare i dati strutturati nel file JSON con le informazioni sui gruppi olfattivi e capacità, in modo tale da allinearlo con le informazioni presenti all'interno del corpus dei dati.

È stata consultata l'ultima versione del file di definizione del vocabolario di Schema.org[8] per verificare se ci fossero delle proprietà specifiche per identificare la fragranza di un oggetto o la capacità. La proprietà più simile alla capacità presente nel vocabolario è `"schema:fuelCapacity"` che, tuttavia, indica la capacità del serbatoio di una automobile. Anche per quanto riguarda le fragranze non sono state trovate proprietà corrispondenti. Per questo motivo, è stata utilizzata `"schema:additionalProperty"`, una proprietà che rappresenta una caratteristica aggiuntiva dell'entità per la quale non esiste una proprietà corrispondente in Schema.org.

Le proprietà sono state aggiunte tramite uno script Python appositamente sviluppato per leggere il file JSON dei prodotti, prelevare le informazioni dalla tabella di output del processo di classificazione con LLM, che contiene i dati su capacità e categorie olfattive, e aggiungerle come proprietà aggiuntive di Schema.org, come mostrato in Figura 4.6. Sono state inoltre aggiornate le categorie nel campo `"category"` per far sì che combacino esattamente con quelle riportate nel corpus dei dati.

[8] `https://schema.org/docs/developers.html`

```
    "additionalProperty": [
        {
            "@type": "PropertyValue",
            "name": "Capacita",
            "value": "75 ml"
        },
        {
            "@type": "PropertyValue",
            "name": "Gruppo Olfattivo",
            "value": "Floreale, Fruttato, Muschiato, Floreale Fruttato, Floreale
            ↪  Muschiato"
        }
    ]
```

Figura 4.6: Esempio delle proprietà aggiuntive di capacità e gruppo olfattivo per un prodotto, salvate all'interno del file JSON dei prodotti.

4.2.5 Completamento del corpus dei dati

Le ultime due caratteristiche dei prodotti da aggiungere alla tabella con le categorie di prodotto, le categorie olfattive e le capacità sono le informazioni sui brand e sui prezzi. Queste caratteristiche sono estratte dal grafo dei prodotti nel file JSON, utilizzando uno script Python dedicato, e vengono salvate in formato tabellare. Attraverso un'operazione di left join con la tabella che includeva già le categorie olfattive, le categorie di prodotto e le capacità, è stato finalizzato il corpus dei dati.

La Figura 4.7 riassume il procedimento completo per la creazione del corpus dei dati con tutte le fasi descritte in precedenza. A partire dalla versione in locale del sito, `ethos.local`, attraverso le operazioni descritte in precedenza, sono state calcolate le informazioni finali dei prodotti raccolte nel corpus dei dati. In blu, sono riportate le caratteristiche dei prodotti che saranno utilizzate per generare le query per la fase di *valutazione* dei sistemi di ricerca.

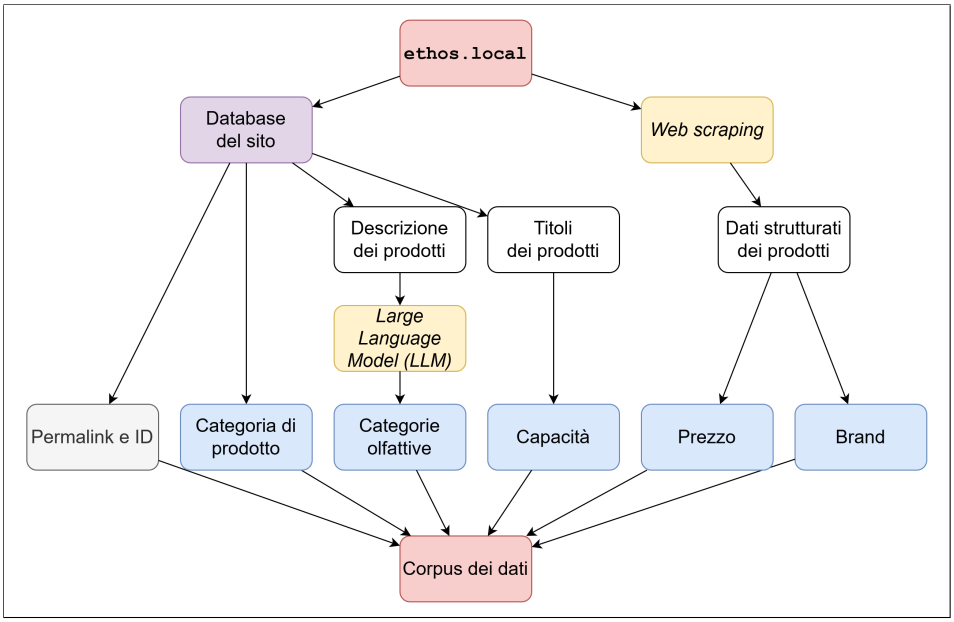

Figura 4.7: Schema che riassume la modalità di creazione del corpus dei dati.

Ogni prodotto, definito da URI e ID sequenziale interno al sito, presenta le seguenti cinque caratteristiche:

- *Brand*;
- *Capacità*, espressa in millilitri;
- *Categoria di prodotto*, ad esempio “Fragranze Donna”;
- *Categorie olfattive*, singole e combinate;
- *Prezzo*, in Euro.

4.3 Ambiente di valutazione

L’impostazione dell’ambiente per la fase di *valutazione* del framework inizia con la creazione della *ground truth*, costituita da un corpus di query di base e i relativi risultati esatti, a partire dal corpus dei dati. Successivamente, sarà necessario ottenere la relativa traduzione delle query in linguaggio naturale e SPARQL. Le

query testuali dovranno poi essere eseguite con il plugin Clerk.io[9], che implementa la ricerca basata su keywords internamente al sito. Le query in linguaggio SPARQL saranno invece eseguite sul grafo di triple RDF serializzato in linguaggio Turtle, utilizzando il server *Apache Jena Fuseki*[10].

I risultati ottenuti da questi due sistemi di ricerca saranno raccolti in un unico file JSON e confrontati con i risultati esatti della ground truth per produrre le metriche quantitative delle prestazioni.

4.3.1 Generazione della ground truth

La query presenti nella ground truth vengono calcolate attraverso la combinazione di tutte le cinque caratteristiche dei prodotti, valutate con ogni valore possibile. Il numero di combinazioni di caratteristiche possibili viene calcolato con l'Equazione 3.1, presentata nel Capitolo 3. In particolare, per il caso di studio vengono calcolate le query con tutti i valori che possono assumere tutte le combinazioni possibili da $N = 2, 3, 4$ caratteristiche. Una query è considerata ammissibile se ha almeno 20 risultati esatti associati.

Per implementare ciò, è stato sviluppato un codice in Python, il quale inizia con il caricamento dei dati sui prodotti dal corpus dei dati in formato tabellare. Successivamente, il programma estrae le capacità più frequenti dei prodotti e identifica i brand con un numero minimo di prodotti, stabilito anch'esso a 20 per potare lo spazio di ricerca da combinazioni che non sarebbero comunque state valide.

I valori possibili considerati nelle query sono i seguenti.

- *Brand*: qualsiasi valore possibile con almeno 20 prodotti associati;
- *Capacità*: i 5 valori con più occorrenze, ovvero $\{30, 50, 100, 150, 200\}$ ml;
- *Categoria di prodotto*: {Fragranze Donna, Fragranze Uomo};
- *Categorie olfattive*: {Agrumato, Ambrato, Aromatico, Chypre, Cuoio, Dolce, Floreale, Fruttato, Gourmand, Legnoso, Muschiato, Senza Profumo, Speziato

[9] `https://www.clerk.io`

[10] `https://jena.apache.org/documentation/fuseki2/`

Leggero, Aromatico Legnoso, Floreale Fruttato, Floreale Legnoso, Agrumato Legnoso, Agrumato Aromatico, Agrumato Fruttato, Agrumato Floreale, Floreale Muschiato, Ambrato Floreale, Agrumato Muschiato};

- *Prezzo*: minori di $\{20, 30, 40, 50, 75, 100\}$ Euro.

Quindi, definita la lista delle caratteristiche disponibili e i relativi valori possibili, il codice utilizza una funzione di filtraggio per elaborare i risultati esatti delle query. Partendo da tutti i prodotti della tabella, ogni caratteristica della query viene usata come filtro rimuovendo i prodotti non corrispondenti alla query. In questo modo, si calcolano i risultati esatti di ogni query. La funzione è anche responsabile della selezione dei risultati dalla tabella in base alle caratteristiche combinate, garantendo che ogni combinazione di valori generata soddisfi il requisito di avere almeno 20 risultati. Al termine dell'elaborazione, tutte le query ammissibili vengono scritte in un file JSON, all'interno delle chiavi `"query"` per la query ritenuta ammissibile e `"true_results"` per la lista di risultati esatti associata alla query.

Alla fine del procedimento, sono state generate il seguente numero di query ammissibili nella ground truth, che saranno alla base dei calcoli delle metriche di valutazione:

- 564 query ammissibili da 2 caratteristiche,
- 368 query ammissibili da 3 caratteristiche,
- 26 query ammissibili da 4 caratteristiche.

Il numero di query ritenute ammissibili diminuisce con l'aumentare del numero di caratteristiche impiegate per la costruzione della query. Questo è dovuto al vincolo di avere almeno 20 risultati esatti associati. Infatti, aggiungendo sempre più caratteristiche, la query diventa sempre più specifica e il numero di risultati associati tende a diminuire progressivamente.

4.3.2 Query testuali

A partire dalle query di base nella ground truth, è necessario ricavare le equivalenti query basate su keywords per la ricerca interna al sito. La conversione da query di

base a query testuale consiste nel riportare i valori assunti dalle cinque caratteristiche, con le stringhe in minuscolo e ordinate come segue. A seconda della presenza o meno dei campi della query, vengono riempiti i campi corrispondenti:

"{categoria} {categoria olfattiva} {capacità} {brand} minori di {prezzo} euro".

Un esempio di conversione da query di base a query testuale è presentato in Figura 4.8. La query originale, nel campo `"query"`, è composta dalle caratteristiche `"category": "Fragranze Donna"`, `"olfactory_category": "Muschiato"` e `"price": "<100"`. Per convertirla in query testuale, si prendono i valori assunti dalle caratteristiche e vengono ordinati, resi minuscoli e, nel caso del prezzo, si rimuove il simbolo <. La query testuale equivalente sarà `"text_query": "fragranze donna muschiato minori di 100 euro"` e potrà essere utilizzata come input per la ricerca basata su parole chiave.

```
"query": {
    "category": "Fragranze Donna",
    "olfactory_category": "Muschiato",
    "price": "<100"
},
"text_query": "fragranze donna muschiato minori di 100 euro",
```

Figura 4.8: Esempio di query di base e query testuale equivalente.

Ottenute le query testuali equivalenti, per eseguire le ricerche in maniera automatica all'interno del sito è stato sviluppato un plugin personalizzato in linguaggio PHP. Il plugin utilizza il sistema di gestione dei contenuti WordPress per invocare una ricerca tramite *WordPress Command Line Interface* (WP-CLI), un'interfaccia a riga di comando per la gestione di installazioni WordPress. Il plugin è eseguito tramite un comando WP-CLI chiamato `export_clerk_query`, che accetta come input la query testuale e restituisce i risultati in forma testuale nel terminale, oltre a salvarli su disco in un file CSV.

Il plugin permette di invocare una ricerca basata su parole chiave all'interno del sito, a partire dalla query testuale passata come argomento del comando. Viene effettuata la ricerca dei prodotti nel database e vengono restituiti gli identificativi dei prodotti corrispondenti. I prodotti restituiti come risultati saranno quelli che

hanno una corrispondenza nel titolo e nella descrizione con tutti i token della query testuale, in seguito ad un'operazione di stemming, come descritto nel Capitolo 3.

L'esecuzione automatica delle query testuali è svolta mediante uno script Python che utilizza il comando WP-CLI del plugin PHP in esecuzione su Docker. Lo script legge il file JSON per prelevare le query di base e, per ciascuna, le traduce in query testuali, chiamando il plugin `export_clerk_query` per eseguire le ricerche. I risultati vengono salvati in un file CSV dal quale gli identificativi dei prodotti trovati vengono estratti. Al termine del processo, i file CSV temporanei vengono eliminati e il file JSON è aggiornato con i campi `"text_query"` e `"text_results"`.

4.3.3 Query in SPARQL

Le query in linguaggio SPARQL sono ottenute mediante una procedura di traduzione a partire dalle query di base. Siccome la query di base dichiara esplicitamente quali caratteristiche sono utilizzate con quali valori richiesti, è bastato adattare la costruzione dell'equivalente query SPARQL in base alla presenza o meno delle cinque caratteristiche.

Anche per la generazione e per la successiva esecuzione delle query in SPARQL è stato sviluppato uno script in linguaggio Python. Il codice legge il campo `"query"` dal file JSON della ground truth e, a seconda della presenza o meno delle cinque caratteristiche, costruisce le clausole necessarie, popolando la *f-string* `sparql_query` presentata nella Figura 4.9. La f-string, o "formatted string literal", è una sintassi introdotta recentemente in Python che permette di includere direttamente espressioni all'interno di una stringa. Sfruttando questa funzionalità, attraverso una serie di funzioni dedicate a ogni caratteristica, vengono generate dinamicamente le clausole SPARQL e quindi la query completa.

```
sparql_query = f"""
PREFIX schema: <http://schema.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?product
WHERE {{
  ?product a schema:Product ;
           schema:name ?name .
  {offer_clause}
  {category_clause}
  {capacity_clause}
  {brand_clause}
  {olfactory_category_clause}
  {price_clause}
}}
"""
```

Figura 4.9: Blocco di codice in Python usato per costruire una query SPARQL dinamica.

Le clausole vengono create in modo condizionale in base alla presenza o meno di una caratteristica. Se è presente una caratteristica, la funzione dedicata inserisce la clausola relativa alla caratteristica, inserendo anche il valore associato ad essa. Se una caratteristica non è presente nella query di base, la corrispondente clausola SPARQL non viene inclusa nella query finale. Quando una query di base specifica un prezzo, il sistema aggiunge automaticamente anche la clausola `schema:offers ?offer`.

Lo script produrrà in output una query in linguaggio SPARQL equivalente alla query originale. Per esempio, la query presentata in Figura 4.10 è stata generata convertendo la query in Figura 4.8. Da notare come vengono utilizzate le proprietà personalizzate con `"schema:additionalProperty"` per la capacità e la categoria olfattiva. Inoltre, il prefisso `xsd` indica il vocabolario XML Schema Definition, utilizzato per garantire l'interpretazione corretta dei tipi di dati in SPARQL.

```
PREFIX schema: <http://schema.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?product
WHERE {
  ?product a schema:Product ;
           schema:name ?name .
  ?product schema:offers ?offer .
  ?product schema:category ?category .
  FILTER(CONTAINS(?category, "Fragranze Donna"))

  ?product schema:additionalProperty [
    a schema:PropertyValue ;
    schema:name "Gruppo Olfattivo" ;
    schema:value ?olfattivo
  ] .
  FILTER(CONTAINS(?olfattivo, "Muschiato"))

  ?offer schema:price ?price ;
         schema:priceCurrency ?currency .
  FILTER(xsd:decimal(?price) < 100)
}
```

Figura 4.10: Query SPARQL equivalente alla query in Figura 4.8.

Per quanto riguarda l'esecuzione delle query in SPARQL, è stato utilizzato il server Apache Jena Fuseki, progettato per la gestione di dati RDF e l'esecuzione di query in linguaggio SPARQL 1.1. È quindi stato codificato uno script in Python per l'esecuzione automatica delle query SPARQL. Lo script esegue le query sul grafo nel file JSON dei prodotti, convertito in linguaggio Turtle, contenente esattamente 176048 triple RDF per descrivere 6240 prodotti.

Lo script automatizza l'esecuzione delle query generate in SPARQL, interfacciandosi con il server Fuseki. La funzione principale dello script è quella di inviare ciascuna query SPARQL, generata dinamicamente, al server Fuseki e di raccogliere i risultati. Il processo inizia caricando le query di base dal file JSON della ground truth. Per ciascuna query, viene costruita la relativa query SPARQL, con il procedimento spiegato in precedenza, e viene inviata al server Fuseki tramite una richiesta HTTP di tipo GET. La risposta viene restituita in formato JSON e contiene gli URI dei prodotti corrispondenti alle query.

Gli URI estratti dai risultati SPARQL vengono successivamente trasformati in

ID di prodotti, grazie a una mappatura tra gli URI dei prodotti e i loro ID, caricata dalla tabella del corpus dei dati. Una volta ottenuti gli ID, vengono aggiunti al file JSON, insieme alla query SPARQL, salvati nei campi `"sparql_query"` e `"sparql_results"`. In questo modo, tutti i risultati delle query SPARQL vengono raccolti e formattati come ID di prodotti, allo stesso modo dei risultati esatti e dei risultati provenienti da query testuali, pronti per essere utilizzati nel calcolo delle prestazioni.

4.4 Calcolo delle metriche quantitative

La Figura 4.11 mostra un esempio di query salvata nel file JSON utilizzato nel caso di studio per memorizzare la ground truth e i risultati ottenuti. La query originale, parte della ground truth, è memorizzata nel campo `"query"`, mentre i relativi risultati esatti sono salvati nella lista `"true_results"`.

La figura rappresenta lo stato del file dopo l'esecuzione delle versioni testuali e SPARQL della query originale, salvate rispettivamente nei campi `"text_query"` e `"sparql_query"`. Per ciascuno degli approcci impiegati, i risultati ottenuti sono memorizzati nelle rispettive liste. Nella lista `"text_results"` ci sono i risultati della query testuale, mentre in `"sparql_results"` ci sono i risultati della query SPARQL. Tutti i prodotti nelle liste di risultati sono identificati sotto forma di ID univoci.

```
{
  "query": {
    "category": "Fragranze Donna",
    "capacity": "100 ml",
    "olfactory_category": "Floreale Fruttato"
  },
  "text_query": "fragranze donna floreale fruttato 100 ml",
  "sparql_query": "PREFIX schema: <http://schema.org/>\n    PREFIX xsd:
  ↪  <http://www.w3.org/2001/XMLSchema#>\n\n    SELECT ... ... ",
  "true_results": [
    22044, 23740, 23752, 27264, 31517, 33283, 33354, 63483, 124613,
    124607, 124583, 124590, 124575, 132500, 132502, 132544, 125705,
    125710, 125090, 133130, 133132
  ],
  "text_results": [
    22037, 22890, 24317, 24462, 25464, 27264, 29267, 29284, 29812,
    31570, 31684, 33443, 33456, 43432, 50705, 53231, 62991, 64109,
    92645, 118664, 118669, 161798, 170960, 186966, 203969
  ],
  "sparql_results": [
    22044, 23740, 23752, 27264, 31517, 33283, 33354, 63483, 124575,
    124583, 124590, 124607, 124613, 125090, 125705, 125710, 132500,
    132502, 132544, 133130, 133132
  ]
}
```

Figura 4.11: Esempio di query nel JSON utilizzato per la valutazione dei risultati.

Poiché tutti i risultati sono formattati come liste di ID, è stato possibile gestirli in modo uniforme per il calcolo delle metriche di valutazione. In particolare, le metriche selezionate sono state le seguenti, come già spiegato in modo più approfondito nella sezione dedicata all'interno del Capitolo 3.

- *Precisione*: tasso di risultati corretti sui risultati restituiti dal sistema,
- *Richiamo*: tasso di risultati corretti restituiti sul totale dei risultati esatti,
- *F1 measure*: media armonica di precisione e richiamo.

Per calcolare le metriche, è stato sviluppato un codice in Python che legge il file JSON dei risultati, nel formato illustrato in Figura 4.11. In particolare, lo script esegue il calcolo delle metriche di precisione e richiamo utilizzando due liste di risultati: una per le query testuali, `"text_results"`, e una per le query SPARQL, `"sparql_results"`. Le due liste vengono confrontate con la lista dei risultati esatti

in `"true_results"`. Per ciascuna lista di risultati di ogni query, si calcola precisione e richiamo. Infine, le metriche vengono aggregate e mediate su tutte le query per ottenere la precisione media e il richiamo medi. I due valori finali sono quindi usati per calcolare la F1 measure. In questo modo, si può fornire una valutazione complessiva delle prestazioni del sistema.

Lo script può, inoltre, calcolare le metriche escludendo le query con una particolare caratteristica, come il prezzo, per consentire un'analisi del comportamento dei due sistemi in relazione alle diverse caratteristiche.

4.5 Da query testuali a query SPARQL

Un approccio alternativo per l'esecuzione di query in linguaggio naturale, proposto in questo lavoro di tesi, consiste nella conversione diretta da linguaggio naturale a SPARQL. L'obiettivo è offrire un metodo diverso dalla tradizionale ricerca per parole chiave, consentendo comunque agli utenti di eseguire interrogazioni in linguaggio naturale, che rimangono gli input principali della ricerca.

Per automatizzare questo processo sullo stesso corpus di query precedentemente calcolato, è stata sfruttata la capacità dei Large Language Models (LLM) di interpretare i token che compongono le query. Il sistema dovrebbe quindi riconoscere automaticamente gli elementi strutturati, scelti tra un insieme predefinito di caratteristiche in base al contesto d'uso previsto. L'idea è stata di riutilizzare il processo di generazione delle query SPARQL precedentemente analizzato, che compila automaticamente le clausole corrispondenti alle caratteristiche identificate tramite una f-string in Python, come mostrato in Figura 4.9.

Dal punto di vista implementativo, è stato sviluppato un codice Python che integra il modello linguistico Llama 3, già utilizzato in precedenza per ricavare le categorie olfattive. Il LLM, in questo caso, è utilizzato per comprendere e mappare le query testuali degli utenti sulle cinque caratteristiche strutturate di `"brand"`, `"category"`, `"capacity"`, `"olfactory category"` e `"price"`. Il sistema inizia definendo un *prompt* che specifica come il modello `Llama-3-Groq-70B-Tool-Use` deve analizzare una query in linguaggio naturale e compilare i campi relativi ai prodotti. Per ogni query, il LLM esporterà una lista di coppie caratteristica, valore.

Successivamente, il codice genera una query in SPARQL basata sui valori estratti dal modello, in modo simile a quanto è stato già svolto in precedenza per tradurre le query di base. Ogni caratteristica, come la categoria del prodotto o la sua capacità, viene inserita in apposite clausole all'interno della query, utilizzando la f-string in Figura 4.9. Una volta generata la query SPARQL, questa viene inviata al server Fuseki per l'esecuzione. La risposta restituisce una lista di URI che identificano i prodotti corrispondenti ai criteri della query, trasformati in ID univoci di prodotti tramite il corpus dei dati.

Il sistema verifica ogni volta che l'output del LLM rispetti il formato richiesto, ripetendo la chiamata al LLM se necessario. Inoltre, vengono applicate eventuali modifiche all'output, come la capitalizzazione dei campi, prima di generare la query SPARQL finale. Infine, il codice elabora e salva i risultati in file JSON aggiornati, mantenendo una traccia delle query LLM e dei risultati associati, completando così l'intero ciclo dall'input in linguaggio naturale alla generazione della query SPARQL e l'ottenimento dei risultati.

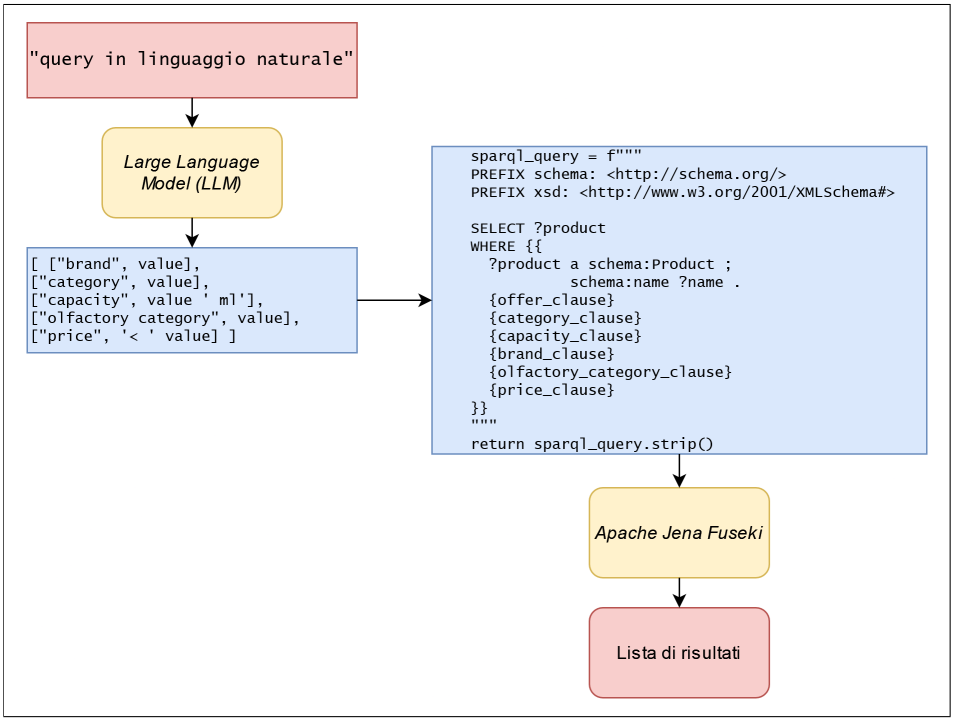

Figura 4.12: Schema che riassume l'approccio utilizzato nel caso di studio per l'esecuzione di query testuali, traducendole in query SPARQL tramite LLM.

Il procedimento è schematizzato in Figura 4.12. La query in linguaggio naturale in input viene elaborata dal Large Language Model che fornisce in output una lista di coppie caratteristica, valore. A seconda delle caratteristiche presenti nella lista, vengono compilate le clausole della query SPARQL che viene eseguita dal server Apache Jena Fuseki. Alla fine del processo, vengono aggiunti al file JSON dei risultati i campi `"llm_sparql"` contenente la query tradotta ed `"llm_results"` con la lista di risultati prodotti, in modo tale da poter calcolare le metriche di valutazione. In Figura 4.13 è mostrato un esempio di traduzione in SPARQL della query testuale della Figura 4.8. Il risultato è equivalente alla query SPARQL generata direttamente dalla ground truth, in Figura 4.10.

```
PREFIX schema: <http://schema.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?product
WHERE {
  ?product a schema:Product ;
           schema:name ?name .
  ?product schema:offers ?offer .
  ?product schema:category ?category .
  FILTER(CONTAINS(?category, "Fragranze Donna"))

  ?product schema:additionalProperty [
    a schema:PropertyValue ;
    schema:name "Gruppo Olfattivo" ;
    schema:value ?olfattivo
  ] .
  FILTER(CONTAINS(?olfattivo, "Muschiato"))

  ?offer schema:price ?price ;
         schema:priceCurrency ?currency .
  FILTER(xsd:decimal(?price) < 100)
}
```

Figura 4.13: Query SPARQL ottenuta dalla traduzione della query testuale nel campo `"text_query"` del codice in Figura 4.8.

Il principale vantaggio di questo processo è la possibilità di scrivere una query in linguaggio naturale, che viene poi elaborata utilizzando i dati strutturati dei prodotti, senza richiedere all'utente una conoscenza pregressa della sintassi del linguaggio SPARQL.

Capitolo 5

Valutazione dei Risultati

Il presente capitolo è dedicato alla valutazione dei risultati prodotti dal caso di studio, misurati con le metriche quantitative di *precisione*, *richiamo* e *F1 measure*. Si parte con la presentazione dei risultati provenienti dal sistema di ricerca basato su parole chiave, con le relative considerazioni. Quindi, vengono presentati e opportunamente commentati i risultati provenienti dall'esecuzione delle query semantiche in SPARQL.

I risultati quantitativi dei due sistemi sono presentati in dettaglio nelle Tabelle 5.1 e 5.2, che mostrano le metriche quantitative calcolate rispettivamente su tutte le query e solo sulle query senza vincoli di prezzo. Quindi, si confrontano i risultati per determinare vantaggi e svantaggi di ciascuno.

Infine, vengono presentati i risultati provenienti dall'esecuzione di query testuali tradotte in query SPARQL tramite l'ausilio di LLMs, comparandola all'esecuzione di query testuali con l'approccio di ricerca basato su parole chiave e al sistema di ricerca che prende in input direttamente le query SPARQL.

5.1 Risultati di query testuali

Di seguito vengono presentati i risultati ottenuti dalla valutazione delle query testuali eseguite sul sistema di ricerca basato su parole chiave. Sono state analizzate diverse combinazioni di caratteristiche, o *features*, per la creazione delle query. In

totale, sono state generate un certo numero di query *ammissibili*, definite come quelle che hanno restituito almeno 20 risultati. I dati raccolti includono il numero di features utilizzate per la costruzione della query, il numero di query ammissibili usate nel calcolo della media e i valori medi di precisione, richiamo e F1 measure.

Per le query composte da 2 caratteristiche casuali tra le cinque disponibili, sono stati ottenuti i valori di precisione, richiamo e F1 measure mediati su 564 query valide. La precisione media è di 0.095 e indica che dei risultati restituiti dal sistema di ricerca basato su parole chiave, in media il 9.5% era effettivamente corretto. Il richiamo medio è di 0.072 e quindi, in media, il sistema è stato in grado di restituire il 7.2% di tutti i risultati corretti. La F1 measure media è invece di 0.082.

Se si vanno a considerare le 368 query composte da 3 caratteristiche, i valori di precisione, richiamo e F1 measure medi decrementano a 0.014, 0.021 e 0.017, fino ad azzerarsi completamente per le 26 query composte da 4 caratteristiche.

I valori ottenuti sono decisamente bassi e indicano una difficoltà evidente del sistema di restituire correttamente i risultati. Inoltre, apparentemente, sembra che con query più articolate, il sistema decrementi le prestazioni fino ad azzerarsi. La Figura 5.1 mostra l'andamento della F1 measure in funzione del numero di caratteristiche che compongono le query. Il grafico mostra che, con query più specifiche, si ottengono prestazioni minori per la ricerca basata su parole chiave.

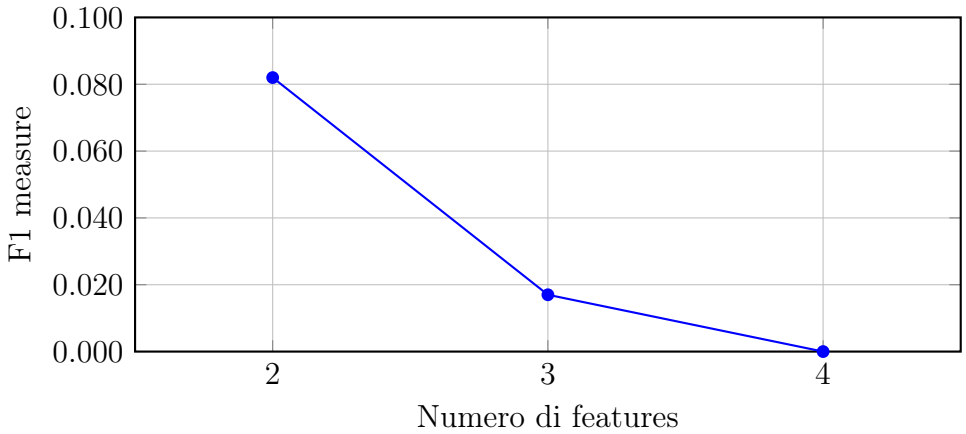

Figura 5.1: Andamento della F1 measure in funzione del numero di features della ricerca basata su parole chiave, calcolata su tutte le query.

Per indagare a fondo nella causa di questa decrescita di prestazioni, sono state valutate le query in base alla presenza o meno di determinate caratteristiche ed è stato rilevato che, per le query in linguaggio naturale che contengono vincoli di prezzo, il sistema non restituisce alcun risultato. Ad esempio, se la query contiene le parole `"minori di 50 euro"`, la lista di risultati associata alla query è vuota. Il sistema deve cercare una corrispondenza con tutti i token che compongono la query e questo vuol dire che non riesce a trovare corrispondenze per le parole `"minori"`, `"di"`, `"50"` ed `"euro"` nella descrizione o nel titolo di alcun prodotto. Quindi, precisione e richiamo per tale query saranno esattamente zero.

Di conseguenza, la decrescita ed eventuale azzeramento delle prestazioni è dovuta a un'alta proporzione di query con vincoli di prezzo che il sistema non riesce a elaborare, sul totale delle query. Per questo motivo, sono state calcolate le metriche unicamente sulle query senza vincoli di prezzo.

In questo caso, le prestazioni sono significativamente diverse. Per le 131 query ammissibili senza vincoli di prezzo composte da 2 caratteristiche, il valore di precisione media è di 0.408 e quindi, in media, il 40.8% dei risultati restituiti dal sistema era corretto. Inoltre, il valore di richiamo medio di 0.312 indica che il sistema è stato in grado di trovare il 31.2% di tutti i risultati corretti.

Per le 27 query ammissibili composte dalla combinazione di 3 caratteristiche, i valori medi di precisione e richiamo scendono rispettivamente a 0.192 e 0.283. Tra le query a 4 caratteristiche, invece, tutte quelle ammissibili presentano vincoli di prezzo, causando i valori uguali a zero per precisione, richiamo e F1 measure descritti in precedenza.

I valori ottenuti da queste metriche indicano che, per le query senza vincoli di prezzo, il sistema riesce parzialmente a restituire risposte corrette, ma non riesce a raggiungere tutte le risposte corrette e un numero significativo di risultati restituiti dal sistema non sono effettivamente corretti.

Per riassumere, il sistema di ricerca basato su parole chiave si è dimostrato decisamente inefficiente nel rispondere alle query testuali. Il sistema presenta particolari difficoltà con le query più specifiche, composte da un numero maggiore di

caratteristiche. È completamente assente da parte del sistema una capacità di interpretazione dei token che compongono la query testuale, che si limita a essere analizzata mediante una corrispondenza all'interno del testo della descrizione o del titolo. A dimostrazione di questo, se una query testuale esprime vincoli sul prezzo, il sistema non restituisce alcun risultato.

5.2 Risultati di query SPARQL

Per tutte le query SPARQL, indipendentemente dal numero di caratteristiche usate per comporle, sono stati ottenuti valori di 1.000 per la precisione media, di 1.000 per il richiamo medio e, conseguentemente, di 1.000 per la F1 measure media. In altre parole, il 100.0% dei risultati restituiti dal sistema è corretto, e il sistema è in grado di restituire il 100.0% di tutti i risultati corretti, sul setup sviluppato per produrre le metriche.

I valori prodotti mostrano una completa capacità del sistema di ricerca semantica di elaborare le informazioni strutturate che compongono i prodotti e utilizzarle per restituire risultati completamente corretti.

Per una completezza di confronto con i dati prodotti dal sistema basato su parole chiave sulle query senza vincoli di prezzo, sono stati calcolati anche per la ricerca semantica i risultati sulle query senza vincoli di prezzo.

Anche in questo caso, i risultati mostrano una capacità da parte del sistema di ricerca semantica di restituire risultati completamente corretti, con valori di precisione, richiamo e F1 measure quasi sempre pari a 1.000 per le query senza vincoli di prezzo composte da 2 e 3 caratteristiche. L'unico valore diverso da 1.000 è rappresentato dal richiamo medio per le query SPARQL senza vincoli di prezzo e composte da 2 feature, che è di 0.999. Questo leggero scostamento è dovuto alla presenza di un prodotto nei risultati esatti di alcune query, e quindi presente nel corpus dei dati, ma non presente nel file JSON-LD utilizzato per eseguire le query SPARQL. Pertanto, il sistema di ricerca non ha potuto considerarlo nei risultati restituiti.

A differenza dell'approccio di ricerca basato su parole chiave, che è privo della

capacità di gestire query con vincoli di prezzo, il sistema di ricerca basato su query SPARQL è perfettamente in grado di elaborarle. Questo è dovuto a due fattori principali: la capacità di SPARQL di effettuare interrogazioni precise su grafi di dati strutturati e la possibilità di specificare criteri complessi attraverso *triple pattern*, come le condizioni sui prezzi, all'interno della query stessa. Il prezzo, nella query SPARQL, è gestito come tipo `xsd:decimal`, il che consente di eseguire operazioni di confronto numerico, come maggiore di, minore di o uguale a, garantendo così una selezione accurata dei risultati.

5.3 Confronto dei risultati

Avendo ottenuto le misure quantitative delle prestazioni dei due sistemi di ricerca, è possibile fare un confronto per analizzare i due sistemi in modo oggettivo. I risultati derivanti dai due sistemi, calcolati su tutte le query, sono presentati all'interno della Tabella 5.1. La tabella mostra rispettivamente il numero di features utilizzate per le query, il numero di query ammissibili su cui è calcolata la media, e i valori medi di precisione, richiamo e F1 measure sia per la ricerca basata su parole chiave che per la ricerca semantica con query SPARQL.

# di features	**Query ammissibili**	**Ricerca basata su parole chiave**			**Ricerca semantica con query SPARQL**		
		Pr.	**Ric.**	**F1**	**Pr.**	**Ric.**	**F1**
2	564	0.095	0.072	0.082	1.000	1.000	1.000
3	368	0.014	0.021	0.017	1.000	1.000	1.000
4	26	0.000	0.000	0.000	1.000	1.000	1.000

Tabella 5.1: Risultati dei due sistemi di ricerca oggetto di confronto, per tutte le query.

La Tabella 5.2 mostra il numero di features che compongono le query, il numero di query ammissibili e le metriche quantitative dei due sistemi, calcolate unicamente sulle query senza vincoli di prezzo.

# di features	**Query ammissibili**	**Ricerca basata su parole chiave**			**Ricerca semantica con query SPARQL**		
		Pr.	**Ric.**	**F1**	**Pr.**	**Ric.**	**F1**
2	131	0.408	0.312	0.353	1.000	0.999	1.000
3	27	0.192	0.283	0.229	1.000	1.000	1.000
4	0	/	/	/	/	/	/

Tabella 5.2: Risultati dei due sistemi di ricerca oggetto di confronto, per query senza prezzo.

La ricerca basata su parole chiave presenta una capacità limitata di gestire query complesse, con le metriche quantitative che mostrano una riduzione delle prestazioni in base al numero di caratteristiche utilizzate nella query, sinonimo di query più specifiche, come mostrato nella Tabella 5.1. Al contrario, il sistema semantico è in grado di gestire correttamente query di qualsiasi tipo, indipendentemente dalla loro specificità o complessità, utilizzando in modo efficace le informazioni strutturate dei prodotti. I valori delle metriche sono infatti tutti 1.000 o vicini ad 1.000 per il sistema di ricerca basato su query SPARQL, a differenza della ricerca basata su parole chiave.

Anche se meno efficace, il principale vantaggio del sistema di ricerca basato su parole chiave risiede nella possibilità di inserire una query testuale, significativamente più semplice da formulare per l'utente rispetto a una query in linguaggio SPARQL, che presenta una sintassi ben definita. Per questo motivo la ricerca semantica, sebbene sia più performante, è meno accessibile. Per garantire prestazioni ottimali, la ricerca semantica deve inoltre essere supportata da dati strutturati formattati correttamente e di buona qualità, da cui ne dipende direttamente. La ricerca basata su parole chiave potrebbe invece essere la soluzione più semplice per quei sistemi in cui non è richiesta un'elevata specificità negli elementi proposti nel catalogo, ed è più flessibile a dati di bassa qualità o non strutturati.

Per combinare il vantaggio del sistema di ricerca basato su parole chiave, che consente l'uso di una qualsiasi query testuale come input, con la capacità della ricerca semantica di produrre risultati completi ed estremamente specifici, è stata sviluppata una soluzione in grado di tradurre automaticamente query testuali in query SPARQL, come descritto nel Capitolo 4. I risultati di questo sistema sono

presentati nella sezione seguente.

5.4 Risultati di query testuali tradotte in SPARQL

L'obiettivo dell'approccio basato sulla traduzione di query testuali in query SPARQL è utilizzare le query testuali, che sono lo stesso input della ricerca basata su parole chiave, in un sistema di ricerca in grado di impiegare dati strutturati, al fine di produrre risultati qualitativamente migliori rispetto a quelli ottenuti con la ricerca basata su parole chiave.

La Tabella 5.3 mostra le metriche quantitative di valutazione dei risultati derivanti dal sistema di ricerca basato sulla traduzione di query testuali in query SPARQL con l'ausilio di LLMs. Anche in questo caso, i valori di precisione, richiamo e F1 measure fanno riferimento alla media sul totale delle query valide, in base al numero di caratteristiche utilizzate per comporle.

# di features	**Query ammissibili**	**Precisione**	**Richiamo**	**F1 measure**
2	564	0.484	0.648	0.554
3	368	0.775	0.827	0.800
4	26	0.885	0.885	0.885

Tabella 5.3: Risultati del sistema di ricerca basato sulla traduzione in SPARQL di query testuali, per tutte le query.

Per tutte le query generate dalla combinazione di 2 caratteristiche, i risultati prodotti dal sistema mostrano una precisione media del 48.4%, indicando che, in media, il 48.4% dei risultati restituiti dal sistema di ricerca era corretto. Il richiamo medio ottenuto è invece di 0.648 e indica che il sistema è stato in grado di restituire, in media, il 64.8% dei risultati corretti totali. Per le query a tre caratteristiche, la precisione e il richiamo medi aumentano a 0.775 e 0.827, per arrivare a 0.885 per entrambe le metriche con le query a quattro caratteristiche.

Si possono confrontare le prestazioni del sistema basato su traduzione con il sistema basato su parole chiave. A partire dalla stessa query testuale, il sistema basato su traduzione è in grado di ottenere risultati significativamente migliori di

precisione, richiamo e F1 measure medi. Anche se si confrontano le prestazioni misurate solamente sulle query senza vincoli di prezzo, il sistema basato sulla traduzione di query testuali presenta risultati decisamente migliori, presentati nella Tabella 5.4.

# di features	**Query ammissibili**	**Precisione**	**Richiamo**	**F1 measure**
2	131	0.671	0.690	0.680
3	27	0.617	0.792	0.694
4	0	/	/	/

Tabella 5.4: Risultati del sistema di ricerca basato sulla traduzione in SPARQL di query testuali, per query senza prezzo.

Se si confrontano le metriche prodotte dal sistema basato su traduzione di query testuali in query SPARQL, presentate nella Tabella 5.3, con i risultati derivanti dall'esecuzione diretta di query SPARQL presentati nella Tabella 5.1, si nota come i valori di precisione, richiamo e F1 measure sono abbastanza inferiori. La qualità dei risultati del sistema basato su traduzione dipende infatti sia dalla qualità dei dati nel catalogo di prodotti, sia dalla capacità del modello linguistico di disambiguare correttamente le query.

Un esempio di problema derivante dalla qualità dei dati si manifesta nel catalogo dei prodotti del sito, dove sono presenti numerosi elementi con il brand impostato erroneamente su `"Profumi Donna"` o `"Profumi Uomo"` (al posto del valore effettivo del brand), valori che dovrebbero invece rappresentare la categoria di prodotto. Quindi, quando il LLM segue le istruzioni del prompt di compilare la lista di caratteristiche, associa il valore `"Profumi Donna"` o `"Profumi Uomo"` alla categoria di prodotto, ragionevolmente. Di conseguenza, la query produrrà dei risultati sbagliati secondo la ground truth, ma in questo caso perché i dati non sono completamente corretti.

A valle di tutte le osservazioni, si può affermare che il sistema di ricerca basato sull'interpretazione con LLM di query testuali e la successiva traduzione in query semantiche in linguaggio SPARQL è decisamente efficace. È un sistema più versatile rispetto alla ricerca basata su parole chiave e si adatta a qualsiasi livello di specificità richiesto da una query. L'utente esegue la query direttamente in forma

testuale, senza alcuna conoscenza pregressa della sintassi di SPARQL, e il sistema riconosce automaticamente il significato dei token della query. Per garantire che il sistema restituisca risultati corretti è necessario che il LLM sia in grado di estrapolare correttamente le caratteristiche richieste dalla query testuale e che i dati che compongono il catalogo su cui eseguire la ricerca siano completi e di buona qualità.

5.5 Discussione dei risultati

L'analisi condotta presenta alcuni dettagli che necessitano di essere esplicitati per garantire una corretta interpretazione dei risultati. Nell'ambito delle annotazioni semantiche, è consueto misurare l'incremento di visite o il *click-through rate* (CTR) nei mesi successivi all'introduzione di tecnologie capaci di elaborare i dati strutturati di una pagina Web, come indicatore di miglioramento [8].

Il framework presentato nel presente lavoro di tesi non usa le metriche menzionate precedentemente a causa della necessità di non modificare la versione in produzione del sito di eCommerce oggetto di studio e di ottenere risultati quantitativi misurabili in termini di *precisione* e *richiamo*, che siano anche riproducibili e ricavabili in tempi ridotti rispetto agli approcci tradizionali.

La metodologia adottata ha richiesto la costruzione di una *ground truth*, dalla quale sono state generate le query testuali e le query SPARQL da eseguire sui due sistemi di ricerca. La rilevanza di un risultato prodotto è stata quindi misurata confrontando una lista predefinita di risultati esatti. Questo modello di confronto non richiede, per la misura delle prestazioni dei sistemi, l'intervento di utenti reali, ad esempio per formulare le query e successivamente identificare i risultati rilevanti. Pertanto, è stata necessaria una metodologia per generare un insieme di query e, allo stesso tempo, per produrre una lista di risultati esatti correlata alla query.

Nonostante i risultati ottenuti dal sistema di ricerca semantica siano significativi, è importante riconoscere che la generazione delle query è stata direttamente influenzata dai dati precedentemente creati. Questo induce una forma di *bias*, manifestata nei valori di 100.0% di precisione e richiamo riportati nella Tabella 5.1, che potrebbero non riflettere accuratamente i valori effettivi.

Idealmente, sarebbe stato utile l'utilizzo di log contenenti query reali degli utenti. Tuttavia, essendo tale informazione indisponibile, si è proceduto con la creazione della ground truth per la valutazione. Sebbene fossimo consapevoli che questa scelta potesse introdurre una forma di bias, essa ha comunque consentito di effettuare una valutazione preliminare in termini di precisione e richiamo.

Di conseguenza, nonostante la metodologia adottata abbia portato a risultati quantitativi robusti e riproducibili, sarebbe interessante confrontarla anche in un contesto reale, coinvolgendo gli utenti, per confermare ulteriormente le conclusioni del presente caso di studio.

Capitolo 6

Conclusioni

Il lavoro di tesi ha riguardato la definizione di un framework per l'annotazione semantica delle risorse in un sito Web e per la valutazione dell'impatto di tali annotazioni sul ritrovamento delle informazioni.

Il framework è stato successivamente applicato a un sito Web di eCommerce attualmente in uso. Dall'applicazione pratica del framework sono state generate metriche quantitative che misurano le prestazioni di un sistema di ricerca basato su parole chiave, interno al sito, e di un sistema di ricerca semantico basato su query in linguaggio SPARQL, che utilizza i dati strutturati dei prodotti.

Inoltre, è stato sviluppato un sistema di traduzione automatica che converte query in linguaggio naturale in query SPARQL, producendo anch'esso risultati quantitativi. Dal confronto dei risultati ottenuti, sono state tratte conclusioni riguardanti l'efficacia dell'utilizzo dei dati strutturati con i diversi sistemi di ricerca.

6.1 Sintesi dei risultati

Il confronto tra il sistema di ricerca basato su parole chiave e il sistema di ricerca semantico basato su query in SPARQL ha rivelato significative differenze in termini di precisione e di capacità di rispondere a query complesse. Nello specifico, le prestazioni possono essere riassunte mediante le metriche quantitative di precisione media, richiamo medio e F1 measure media calcolate. Ulteriori considerazioni vanno

svolte però anche sulla facilità d'uso e la flessibilità dei sistemi di ricerca, e su quanto siano dipendenti dalla conoscenza pregressa degli utilizzatori e dalla qualità dei dati a disposizione.

Considerando le *prestazioni* complessive dei sistemi, il sistema di ricerca semantico ha prodotto valori assoluti di precisione e richiamo, entrambi pari al 100.0%, e di conseguenza anche la F1 measure ha raggiunto il valore massimo. Il sistema di ricerca basato sulla corrispondenza delle parole chiave ha mostrato, nel migliore dei casi, valori di precisione e richiamo rispettivamente del 40.8% e del 31.2% per le query a 2 caratteristiche, escludendo i vincoli di prezzo. Le prestazioni del sistema peggiorano con l'aumentare del numero di caratteristiche richiesto, e quindi della specificità dell'interrogazione. Il sistema, inoltre, ha prestazioni molto limitate nel caso di query complesse, come query che chiedono tutti i prodotti al di sotto di una certa soglia di prezzo, a causa della natura stessa del sistema. In particolare, per tali query, il sistema non è in grado di restituire alcun risultato.

In termini di *flessibilità*, l'accuratezza della ricerca semantica è fortemente dipendente dal modo in cui sono strutturati i dati e nel modo in cui le query sono state implementate. Ad esempio, per alcune caratteristiche come le categorie olfattive, si cerca una corrispondenza esatta tra i valori richiesti e i dati strutturati. Per questo motivo, è di primaria importanza anche che i dati sui quali viene svolta la ricerca semantica siano di buona qualità e completi. D'altra parte, la ricerca basata su parole chiave prevede una serie di operazioni sui token che compongono la query testuale, quali lo stemming, che permette al sistema di essere più flessibile a parole derivate o forme alternative.

Anche se si considera l'*usabilità*, il sistema basato sulla ricerca semantica è decisamente dipendente da una conoscenza pregressa sulla sintassi del linguaggio SPARQL dei suoi utilizzatori per poter formulare query in modo efficace. Inoltre, per poter formulare una query in SPARQL, gli utenti dovrebbero essere a conoscenza del modo in cui sono annotati i dati. Il sistema di ricerca basato su parole chiave è sicuramente più facile da usare, in quanto richiede in input una query testuale formulabile in modo più intuitivo dagli utilizzatori del sistema.

Il terzo sistema analizzato, che utilizza query testuali come input per la ricerca, ma svolge la ricerca traducendo la query in linguaggio SPARQL, ha prodotto

risultati interessanti. Questo sistema combina la facilità d'uso della ricerca basata su keywords con la capacità di elaborare query complesse della ricerca semantica. Infatti, utilizzando le stesse query testuali impiegate nella valutazione della ricerca basata su parole chiave, il sistema ha prodotto valori medi di F1 measure di 55.4%, 80.0% e 88.5%, rispettivamente per le query composte da 2, 3 e 4 caratteristiche. Le prestazioni del sistema sono dipendenti dalla capacità del LLM impiegato nel processo di disambiguare ed interpretare correttamente la query testuale, e dalla qualità dei dati sui quali viene svolta la ricerca.

6.2 Sviluppi futuri

Un possibile sviluppo futuro del sistema di ricerca proposto in questo lavoro di tesi è di integrarlo in un contesto operativo reale, con l'obiettivo di verificarne l'efficacia in situazioni pratiche di utilizzo. Si potrebbe inserire il sistema come metodo di ricerca principale all'interno di siti Web di eCommerce simili al sito analizzato nel caso di studio, e valutare il comportamento del sistema attraverso dati empirici, come un eventuale incremento del numero di visite o del click-through rate misurato a seguito dell'integrazione del sistema. Attraverso il feedback degli utenti, si potrebbero identificare ulteriori aree di miglioramento per produrre un sistema sempre più capace di rispondere alle necessità dei suoi utilizzatori.

Un altro sviluppo interessante riguarda l'applicazione di tecniche di *stemming* anche per quanto riguarda le query strutturate in linguaggio SPARQL, in modo analogo a come avviene nella ricerca basata su parole chiave analizzata. Attualmente, nel modo in cui sono formulate le query semantiche all'interno del caso di studio, si cerca una corrispondenza esatta tra i valori degli elementi della query SPARQL e i dati strutturati che compongono il prodotto. Per esempio, se consideriamo il modello che traduce una query testuale in una query semantica, quando scriviamo in input `"fragranze donna agrumate"`, il LLM potrebbe inserire `"agrumate"` come valore della caratteristica "categoria olfattiva", mentre tutti i valori nel dataset sono scritti in maschile singolare. Di conseguenza, il modello potrebbe non restituire tutte le risposte corrette. Per questo motivo, se applichiamo tecniche di *stemming* con le query SPARQL, si può aumentare la *flessibilità* del modello anche a forme coniugate e derivate delle parole.

Un'ultima considerazione per gli sviluppi futuri per il modello sviluppato potrebbe prevedere l'implementazione di un meccanismo di *ordinamento per rilevanza* dei risultati prodotti. I sistemi di ricerca analizzati nel presente lavoro di tesi prevedono una rilevanza dicotomica, in cui un elemento è rilevante se soddisfa completamente i criteri della query, oppure è considerato non rilevante. Per questo motivo, per query molto specifiche potrebbero non esserci elementi corrispondenti e quindi i sistemi analizzati non restituirebbero alcun risultato. Con una *rilevanza graduata*, con valori reali nell'intervallo [0,1], sarebbe possibile catturare anche la rilevanza parziale degli elementi, ordinandoli dal più rilevante al meno rilevante. Inoltre, si potrebbero utilizzare metriche quantitative più avanzate per la valutazione dei risultati, come la *Mean Average Precision* (MAP), che tiene conto dell'ordinamento dei risultati e della loro rilevanza graduata.

Bibliografia

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, Cambridge, 2 ed., 2007.

[2] M. R. Baye, B. De los Santos, and M. R. Wildenbeest, *Search Engine Optimization: What Drives Organic Traffic to Retail Sites?*, Journal of Economics & Management Strategy, 25 (2016), pp. 6–31.

[3] T. Berners-Lee and D. Connolly, *RFC 1866: Hypertext Markup Language – 2.0*, Tech. Rep. RFC 1866, Internet Engineering Task Force (IETF), 1995. `http://www.w3.org/MarkUp/html-spec/html-spec_toc.html` Accessed: 2024-08-13.

[4] T. Berners-Lee, J. Hendler, and O. Lassila, *The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities*, ScientificAmerican.com, (2001).

[5] D. Brickley and R. Guha, *RDF Schema 1.1*, February 2014. W3C Recommendation.

[6] eBay, *Structured Data - Updates for Fall 2016*. `https://pages.ebay.com/sellerinformation/news/fallupdate16/structured-data.html#tab=whats-new`, 2016. Accessed: 2024-08-14.

[7] Google, *We Knew the Web Was Big*. `https://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`, 2008. Accessed: 2024-08-12.

[8] Google, *Eventbrite boosted traffic 100% with the events search experience on Google*. `https://developers.google.com/search/case-studies/`

eventbrite-case-study?hl=en, 2018. Published on May 8, 2018. Accessed: 2024-08-16.

[9] Google, *A guide to Google Search ranking systems*. https://developers.google.com/search/docs/appearance/ranking-systems-guide?hl=en, 2024. Accessed: 2024-08-16.

[10] Google, *Generate structured data with JavaScript*. https://developers.google.com/search/docs/appearance/structured-data/generate-structured-data-with-javascript, 2024. Last updated: 2024-10-01 UTC. Accessed: 2024-10-04.

[11] Google, *Introduction to structured data markup in Google Search*. https://developers.google.com/search/docs/appearance/structured-data/intro-structured-data?hl=en, 2024. Accessed: 2024-08-16.

[12] R. V. Guha, D. Brickley, and S. Macbeth, *Schema.org: Evolution of Structured Data on the Web*, Commun. ACM, 59 (2016), p. 44–51.

[13] R. Guns, *Tracing the origins of the Semantic Web*, Journal of the American Society for Information Science and Technology, 64 (2013), pp. 2173–2181.

[14] P. Hayes and B. McBride, *RDF Semantics*, W3C Recommendation, World Wide Web Consortium (W3C), February 2004. Editor: Patrick Hayes (IHMC); Series Editor: Brian McBride (Hewlett Packard Labs). Latest Version: http://www.w3.org/TR/rdf-mt/. Previous Version: http://www.w3.org/TR/2003/PR-rdf-mt-20031215/.

[15] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, *OWL 2 Web Ontology Language: Primer (Second Edition)*, December 2012. W3C Recommendation.

[16] A. Iliadis, A. Acker, W. Stevens, and B. Kavakli, *One schema to rule them all: How Schema.org models the world of search*, Journal of the Association for Information Science and Technology, (2023). Available at SSRN: https://ssrn.com/abstract=4345169.

[17] G. Kellogg, P.-A. Champin, D. Longley, M. Sporny, M. Lanthaler,

and N. Lindström, *JSON-LD 1.1: A JSON-based Serialization for Linked Data*, July 2020. W3C Recommendation.

[18] S. Kemp, *Digital 2024 July Global Statshot Report*. https://datareportal.com/reports/digital-2024-july-global-statshot, 2024. Published on July 31, 2024. Accessed: 2024-09-06.

[19] R. Lamers, O. Kilani, B. Greengus, M. Kandel, D. Elkins, D. Padsumbia, G. Sherry, J. Ross, N. Aklecha, and S. Chaudhary, *Introducing Llama-3-Groq-Tool-Use Models*, Groq, (2023). Open-source Models for Advanced Tool Use.

[20] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, *A Comprehensive Overview of Large Language Models*, 2024.

[21] E. Oren, K. Möller, S. Scerri, S. Handschuh, and M. Sintek, *What are Semantic Annotations?*, Digital Enterprise Research Institute, National University of Ireland, Galway, (2006).

[22] L. Page and S. Brin, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. http://infolab.stanford.edu/~backrub/google.html, 1996. Accessed: 2024-08-12.

[23] E. Prud'hommeaux and A. Seaborne, *SPARQL Query Language for RDF*. https://www.w3.org/2001/sw/DataAccess/rq23/, 2006. W3C Editors' Draft, *Revision* : 1.692 of *Date* : 2006/07/03.

[24] K. Rafes, *Regime entailment basic*. https://commons.wikimedia.org/wiki/File:Regime_entailment_basic.svg, 2011. Licensed under CC BY-SA 3.0, via Wikimedia Commons. Available at: https://creativecommons.org/licenses/by-sa/3.0/.

[25] Schema.org, *Getting Started with schema.org using Microdata*. https://schema.org/docs/gs.html.

[26] Schema.org, *Schema.org Website Main Page*. https://schema.org. Version 27.02, released 2024-07-01.

[27] N. Shadbolt, T. Berners-Lee, and W. Hall, *The Semantic Web Revisited*, IEEE Intelligent Systems, 21 (2006), pp. 96–101.

[28] D. Snyder, *How Structured Data Changed Everything at eBay*. `https://en.ryte.com/magazine/structured-data-changed-everything-ebay/`, May 2020. Accessed: 2024-08-14.

[29] M. Sporny, S. McCarron, B. Adida, M. Birbeck, G. Kellogg, I. Herman, and S. Pemberton, *HTML+RDFa 1.1 - Second Edition: Support for RDFa in HTML4 and HTML5*, March 2015. W3C Recommendation 17 March 2015.

[30] L. Sullivan, *Google Estimates More Than 130 Trillion Web Pages*. `https://www.mediapost.com/publications/article/289019/google-estimates-more-than-130-trillion-web-pag.html`, 2016. Accessed: 2024-08-12.

[31] A. van den Bosch, T. Bogers, and M. de Kunder, *Estimating search engine index size variability: a 9-year longitudinal study*, Scientometrics, 107 (2016), pp. 839–856.

[32] W3C SPARQL Working Group, *SPARQL 1.1 Overview*, March 2013. W3C Recommendation.

[33] WHATWG, *Microdata*. `https://html.spec.whatwg.org/multipage/microdata.html`, Sept. 2024. HTML Living Standard — Last Updated 9 September 2024.