1. INTRODUZIONE

- problema e motivazioni del lavoro

- Necessità di una diagnosi precoce per le malattie neurologiche tra cui l'Alzheimer.

- Gestione di grandi moli di dati (genetici, clinici, neuropsicologici)

- Database pubblici: AddNeuroMed e ADNI

- Analisi dati

- Semantica formale dei dati

- Navigazione dati e interrogazioni

- Integrazione fra più sorgenti dati

- Lavoro realizzato

- Ontologia computazionale relativa dominio dei test neuropsicologici per l'osservazione della malattia di Alzheimer

- <IL RESTO LO INDIVIDUIAMO PIU' AVANTI>

2. DATABASE PUBBLICI SULL'ALZHEIMER

2.1 Database pubblici: AddNeuroMed e ADNI

2.2 Descrizione generale di ADNI:

2.2.1 Tipo di dati raccolti

2.2.2 Organizzazione dei dati

2.2.3 Punti deboli relativi all'organizzazione dati.

3. MODELLAZIONE ONTOLOGICA

3.1 Cosa è un'ontologia

3.2 Perché un'ontologia

3.3 Il linguaggio RDF

3.3.1 Dettaglio del linguaggio RDF

3.3.2 RDF e XML

3.4 Il linguaggio OWL

3.4.1 Le classi OWL

3.4.2 Gli individui OWL

3.4.3 Le propretà in OWL

3.4.4 Combinazione di più ontologie in OWL

3.5 Strumenti per l'editing: Protege

- Strumenti per la persistenza: panoramica sui triple store

- Apache TDB

- API per la manipolzione di ontologie: Jena API

- Linguaggi per l'interrogazione: SPARQL

4. SVILUPPO DELL'ONTOLOGIA RELATIVA AI TEST NEUROPSICOLOGICI

4.1 Approccio e metodologia nel trattamento dei dati

4.2 Sviluppo della TBox, Terminological Box

4.3 Inserimento degli individui la ABox, Assertion Box

5. SPERIMENTAZIONE

- Ancora da definire bene, ma l'idea è quella di definire delle query in SPARQL. Ad esempio query che senza ontologia sarebbe più complicato realizzare.

6. CONCLUSIONI

CAPITOLO 2: Database pubblici sull’Alzheimer

Obiettivo di questo capitolo è quello di introdurre alcuni dei database pubblici contenenti dati relativi all’osservazione di soggetti che partecipano a studi su malattie neurodegenerative, e sull’Alzheimer nello specifico. In particolare, vengono presentati due database, che sono il risultato di due iniziative differenti: AddNeuroMed ed ADNI, e che rappresentano, rispettivamente, un’iniziativa europea ed americana.

Questi due studi *multi-centro*, nei quali quindi i dati sono raccolti in numerosi centri medici con metodologie condivise, hanno come scopo quello di collezionare dati al fine di: (i) trovare e valutare *marcatori* per una diagnosi precoce dell’Alzheimer; (ii) monitorare la progressione della malattia; (iii) identificare gli effetti di terapie farmaceutiche specifiche per l’Alzheimer.

Nel corso del capitolo, vengono quindi descritte le caratteristiche principali dei due progetti, per poi presentare le tipologie di dati che sono memorizzate nei rispettivi database.

2.1. AddNeuromed

AddNeuroMed è un’iniziativa europea, nata dalla collaborazione di enti pubblici e privati nell’ambito di un progetto finanziato dal VI Programma Quadro della Commissione Europea, e con il supporto di EFPIA[[1]](#footnote-1), la Federazione Europea delle Industrie ed Associazioni Farmaceutiche, che rappresenta le industrie europee che si occupano di ricerca farmaceutica in Europa. AddNeuroMed è inoltre stato un precursore dell’IMI[[2]](#footnote-2) (Innovative Medicines Initiative), una collaborazione tra l’Unione Europea e l’industria farmaceutica europea.

AddNeuroMed è uno studio multi-centro, durante il quale, il primo soggetto è stato arruolato nel gennaio del 2006 e l’ultimo nel Febbraio del 2008. I dati sono stati raccolti in 6 centri, tra cui anche l’Università degli Studi di Perugia. Per ogni soggetto coinvolto, il protocollo dello studio prevede una visita iniziale, durante la quale viene formulata la diagnosi relativa all’Alzheimer (baseline assessment), una visita di controllo (follow up) ogni 3 mesi per il primo anno, ed una visita annuale per gli anni successivi. Durante le visite di controllo, viene emessa una nuova diagnosi. Lo studio coinvolge un totale di 781 soggetti. Di questi, al momento del baseline assessment, per 258 è stata diagnosticata la malattia di Alzheimer (AD, Alzheimer Disease), per 257 è stato diagnosticato un deterioramento cognitivo lieve (MCI, Mild Cognitive Impairement), mentre per 266 non è stato diagnosticato alcun deterioramento cognitivo significativo (soggetti di controllo).

I dati raccolti in AddNeuroMed appartengono alle seguenti categorie[[3]](#footnote-3):

* **Clinical**: questi dati rappresentano soprattutto il risultato di test neuropsicologici, e cioè quei dati che sono utilizzati per determinare la diagnosi al termine di ogni visita.
* **DNA**: questi dati rappresentano dati genetici <DA COMPLETARE>
* **Imaging**: questi dati sono relativi alle risonanze magnetiche (MRI) effettuate sul cervello. Vengono memorizzati, sia le immagini che rappresentano il risultato della risonanza magnetica, sia il risultato di analisi di tali immagini.
* **Proteomics**: <DA COMPLETARE>
* **mRNA**: <DA COMPLETARE>

Il portale di AddNeuroMed (vedi Figura 1) è disegnato per consentire un rapido accesso ai dati ed ai risultati analitici ottenuti. L’uso dei dati è incoraggiato e non è richiesta la condivisione di eventuali risultati ottenuti. Il portale attraverso il quale poter acedere ai dati è stato sviluppato utilizzando Synapse software system. I contenuti sono fruibili in maniera molto simile ad un file system. I file contenenti i dati possono essere scaricati direttamente o utilizzando un client R, Java, o Python. Si possono consultare i dati anche ad alto livello, utilizzando le master table.

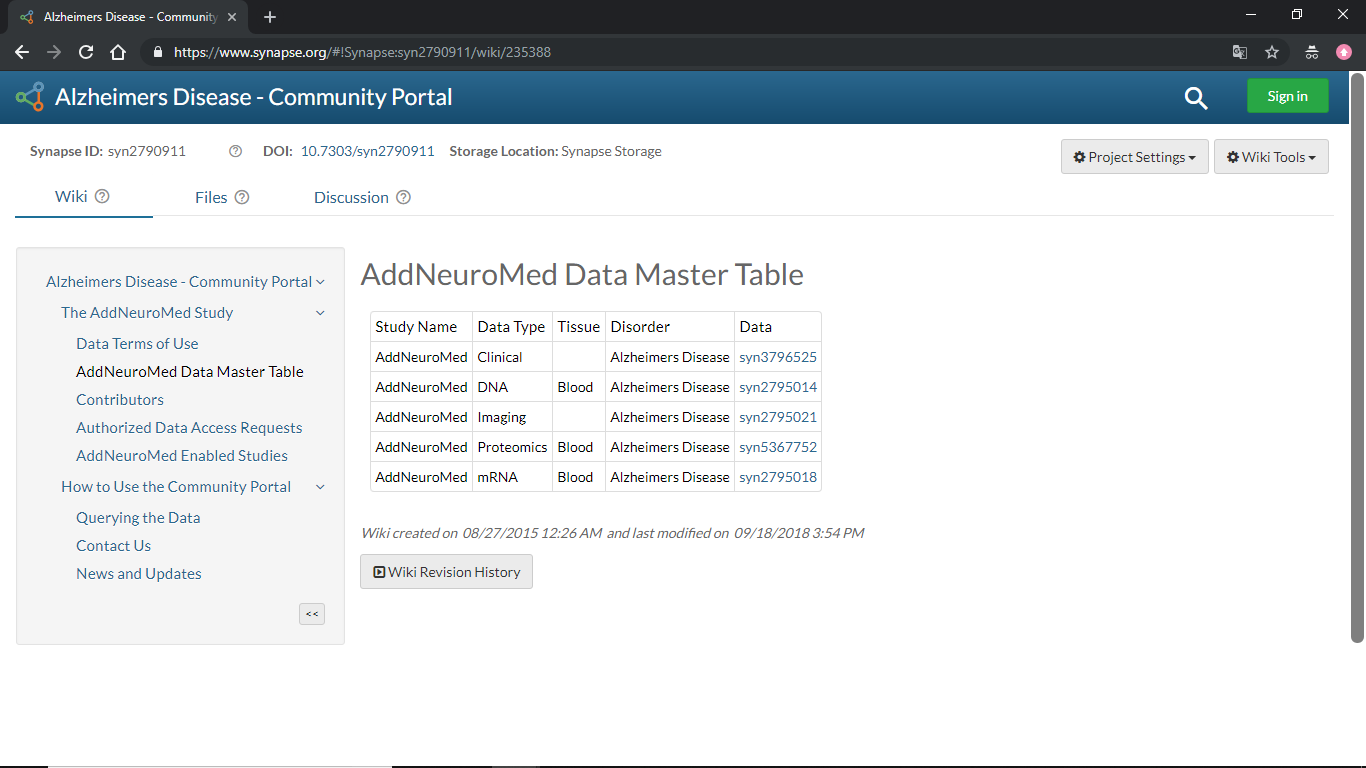


Figura 1 - Portale Web per l'accesso ai dati di AddNeuromed

2.2. ADNI

ADNI (Alzheimer Disease Neuroimaging Initiative) è un’iniziativa sviluppata nel Nord America. Come nel caso di AddNeuroMed, anche ADNI è uno studio multi-centro per la raccolta ed analisi di dati relativi all’osservazione di soggetti nell’ambito dell’Alzheimer. Il progetto ADNI ha inizio nel 2004, attraverso una collaborazione tra pubblico e privato con la partecipazione della Foundation for the National Institutes of Health e del National Institute on Aging. Da allora, il progetto, ha beneficiato di circa 200 milioni di dollari di fondi, il che fa comprendere le dimensioni del progetto stesso. Infatti, i centri coinvolti sono 63 sparsi tra gli Stati Uniti ed il Canada, mentre i soggetti osservati sono in tutto circa 2000. Allo stato attuale, il database gestito da ADNI rappresenta la più grande banca dati sull’Alzheimer disponibile pubblicamente.

Le attività del progetto ADNI si sono susseguite in 4 fasi identificate come ADNI-1, ADNI-GO, ADNI-2 e ADNI-3.

* **ADNI-1** rappresenta la prima fase, e si svolge tra il 2004 e il 2009. In questa fase l’obiettivo era quello di raccogliere dati sui biomarkers e risonanze magnetiche. In questa prima fase i soggetti osservati, e per i quali sono stati raccolti i dati, sono stati circa 800.
* **ADNI-GO** ha inizio nel 2009 e termina nel 2011. L’attenzione in questa fase è focalizzata sui biomarkers, quando la malattia è ancora in stato embrionale. In questa fase ci si avvale anche dei miglioramenti sulle analisi del cervello, grazie all’MRI (Magnetic Resonance Imaging). In questa fase sono entrati a far parte del progetto altri 200 soggetti.
* **ADNI-2**, iniziata nel 2011 e terminata nel 2016. Durante questa fase, sono stati arruolati circa 550 nuovi soggetti, e si è dato seguito ai partecipanti delle precedenti fasi. Inoltre, i livelli di classificazione della diagnosi sono stati ulteriormente raffinati, in quanto sono stati introdotti, due nuovi livelli: early mild cognitive impairment (EMCI) e late mild cognitive impairment (LMCI).
* **ADNI-3**, ha inizio nel 2016 ed è tutt’ora in corso. In questa fase l’enfasi si è spostata sulla ricerca di una correlazione tra i vari dati tra loro eterogenei: dati clinici, genetici, biomarkers ed immagini.

La raccolta dei dati in ADNI ha lo scopo di monitorare l’andamento della malattia. Così la diagnosi potrebbe cambiare in vista di eventuali miglioramenti o peggioramenti delle capacità cognitive e intellettuali del paziente. In questa attività svolge un ruolo centrale il concetto di visita. E’ infatti durante una visita che il paziente viene sottoposto ai test. Il medico può monitorare l’andamento della malattia osservando la cronologia delle visite. Le visite vengono effettuate ogni 6 mesi. Non tutti i test vengono effettuati durante la visita. Il tipo di test che vengono eseguiti sono definiti in un protocollo condiviso da tutti i centri afferenti al progetto, così da avere un primo livello di omogeneità dei dati.

### 2.2.1. Organizzazione dei dati

Il database di ADNI è costituito da decine di tabelle contenenti dati in forma anonima e raggruppate secondo le seguenti categorie:

* **Subject Characteristics**: riporta per ogni soggetto dati anagrafici, quali ad esempio la data di nascita ed il sesso.
* **Medical History**: riporta dati relativi ad eventuali medicinali assunti ed eventuali malattie avute da ogni singolo soggetto.
* **Assessment**: riporta per ogni soggetto osservato e per ogni visita effettuata, i dati relativi ai test neuropsicologici ed i dati relativi alla diagnosi emessa.
* **Biospecimen**: riporta i dati relativi ad esami su campioni di materiale organico, come ad esempio, urina e sangue.
* **Genetic**: riporta, per ogni soggetto, dati relativi alla *genotipizzazione* (genotyping), ovvero al processo capace di determinare quali variazioni genetiche sono proprie di un individuo, e dati relativi al *sequenziamento* (sequencing) cioè al metodo capace di determinare l’esatta sequenza di una certa lunghezza di DNA.
* **Imaging**: riporta i dati relativi all’acquisizione ed analisi di immagini ottenute tramite *risonanza magnetica* (magnetic resonance, MR) e *tomografia ad emissione di positroni* (Positron Emission Tomography, PET).
* **Enrollment**: riporta i dati relativi alle modalità di coinvolgimento nel progetto ADNI di ogni singolo soggetto. Ad esempio, se un soggetto intende donare il proprio cervello per motivi di ricerca alla sua morte.

Per questo lavoro di tesi, l’attenzione è stata riposta principalmente sui dati relativi ai test neuropsicologici. La loro importanza è fondamentale, in quanto i risultati dei test neuropsicologici sono quelli che determinano la diagnosi emessa per ogni soggetto. Questi test consistono in una serie di questionari e misure standardizzati a livello internazionale. Questi test rappresentano esami non invasivi , ovvero senza il prelievo di tessuti o liquidi biologici e senza l’utilizzo di risonanze magnetiche. In generale questi test sono una serie di domande effettuate al paziente e ai suoi familiari. In questo ambito abbiamo:

* **CDR** (*Clinical Dementia Rating*): una scala numerica che ci offre una misura del grado di demenza. Nel caso di ADNI tale scala è la somma di punteggi ottenuti in seguito ad un’intervista che riguarda sei ambiti della vita quotidiana: gli affari comunitari (lavoro, shopping, gruppi sociali), la gestione della casa e degli hobby, il giudizio e la soluzione dei problemi, la cura personale, l’orientamento e la memoria.
* **MMSE** (*Mini-Mental State Examination*): consiste in un questionario di 30 domande che riguardano eventuali disordini, deficit intellettuali e cognitivi.
* **FAQ** (*Functional Assessment Questionnaire*): In questo caso le domande sono 10 e mirano a valutare le capacità funzionali del soggetto intervistato. In particolare la preparazione di cibo e bevande, lo shopping, la gestione delle finanze, la compilazione di moduli, l’uso della televisione, la memoria, i viaggi, gli eventi e i giochi. Come nel caso delle altre tipologie di test, anche per il FAQ, viene ricavato un punteggio finale come somma dei punteggi ottenuti in corrispondenza delle singole domande.
* GDS (Geriatric Depression Scale): una serie di 14 dommande che hanno come possibili risposte Yes o No. Lo score finale in questo caso viene calcolato come il numero di domande che hanno avuto come risposta Yes.
* NPI (Neuropsychiatric Inventory): una serie di 16 domande le cui risposte possono essere o su una scala. In particolare sono presenti delle prove con unna scala da 0 a 5, altre da 1 a 3 e altre ancora da 1 a 4. Altre domande invece hanno possibile risposta Yes/No. Infine una domanda sul disordine del sonno ha come possibile risposta un valore tra 0 e 12. Le domande riguardano in prevalenza il tema dei disordini alimentari, 13 su 16, le altre 3 invece riguardano i disordini legati al sonno.
* NPI-Q (Neuropsychiatric Inventory Questionnaire): in questo caso per ogni prova abbiamo due risposte. Una Yes/No e l’altra invece su una scala da 1 a 3. Le domande riguardano possibili alterazioni del quadro psichiatrico del paziente, ad esempio la presenza di allucinazioni, di irritabilità, di difficoltà motorie ecc. Per ogni domanda viene risposto si o no alla presenza del sintomo, e se il sintomo è presente, si risponde su una scala dando un idea della sua severità. I sintomi considerati sono 12 e alla fine viene calcolato uno score totale.

**<DA INSERIRE L’MMAGINE RELATIVA AD UN QUESTIONARIO TRA QUELLI SOPRA DESCRITTI. CERCARE SU INTERNET>**

### 2.2.2. Accesso ai dati

E’ possibile accedere ai dati direttamente dal portale ADNI (<http://adni.loni.usc.edu/data-samples/access-data/>). I dati sono suddivisi in tre macro aree: *study data*, *image collections* e *genetic data*:

A screenshot of a cell phone

Description automatically generated

Nella sezione dei dati genetici vengono distribuiti dei file VCF contenenti i dati genetici dei soggetti. La sezione è simile a quella degli study data:

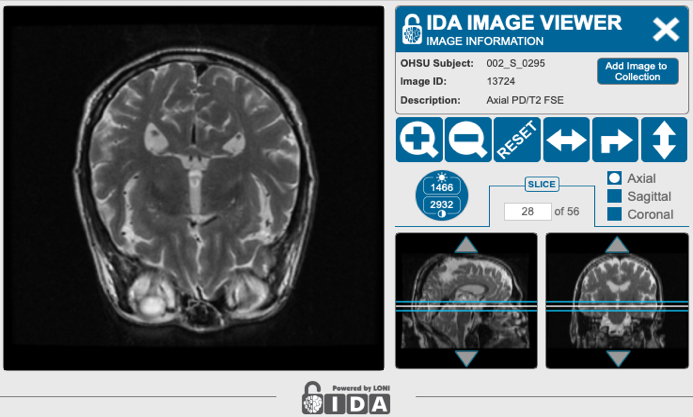
A screenshot of a cell phone

Description automatically generated

E’ stata raccolta l’intera mappatura genetica di 818 partecipanti al programma ADNI. [2]

L’interfaccia di gestione delle immagini è diversa rispetto a quella dei dati genetici e degli study data. E permette una visualizzazione delle immagini direttamente dall’applicazione web:

A screenshot of a social media post

Description automatically generated

Abbiamo una maschera di ricerca, dai i cui risultati possiamo aprire un visualizzatore che ci mette a disposizione varie funzioni di visualizzazione.

Limitandoci ai file che hanno interessato questa tesi, sono presenti nella macroarea degli study data. Dove, sulla spalla sinistra del sito, è presente il menù che ci permette di navigare nelle varie sezioni.

A screenshot of a cell phone

Description automatically generated

Una volta selezionata una voce, a destra abbiamo la lista degli item che possiamo scaricare:

A screenshot of a cell phone

Description automatically generated

Tutti i dati vengono trasferiti alla base di dati ADNI su canale sicuro, secondo degli standard implementati dal nucleo informatico di ADNI stesso. Prima di essere pubblicati subiscono una serie di controlli di qualità. Inoltre è implementata una politica di replica dei dati su diversi nodi, per rendere più sicura la persistenza, e più rapido il ripristino in caso di disaster recovery. [2]

I dati che abbiamo preso in considerazione vengono distribuiti come raw data, in particolare file csv. Contengono i dati clinici di alcuni test: FAQ, MMSE, CDR, GDS, NPI e NPI-Q. I soggetti che si sono sottoposti ai test sono identificati con un valore numerico all’interno dei file.

CAPITOLO 3: Modellazione Ontologica

3.1 Che cosa è un Ontologia

Nei sistemi dove sono diversi i soggetti che intervengono ed interagiscono tra loro, e la mole dei dati è rilevante, si presenta l’esigenza di condividere una semantica formale, così da ridurre significativamente le ambiguità proprie del linguaggio naturale. In particolare, in alcuni domini specifici, come può essere quello medico, è utile costruire micro-linguaggi, partendo dai termini maggiormente utilizzati. A questo scopo il web rappresenta una potenziale sorgente di materia prima, ovvero di dati.

Inoltre, i dati raccolti nell’epoca di Internet, hanno raggiunto dimensioni enormi, tanto che oggi frequentemente si parla di *Big Data*. Un termine diventato di uso comune. Gran parte dei dati memorizzati però è inutilizzato o sottoutilizzato. Scollegato e orientato alla comprensione umana e non delle macchine. Se si vogliono effettuare delle interrogazioni, ai fini di ottenere informazione da dati “grezzi” (*raw data*), le difficoltà sono spesso enormi e scoraggiano coloro che vogliono cimentarsi in questa impresa. Grazie alle ontologie è possibile dare una semantica ai dati, esplicitando in maniera formale il loro significato e le loro relazioni. A questo punto grazie a linguaggi di query, come SPARQL, diventa accessibile la possibilità di estrarre informazione.

Lo sforzo della costruzione di una ontologia, è quello di dare una rappresentazione formale, in maniera tale da poter condividere la conoscenza tra soggetti diversi e consentire ad una macchina di poter processare e ragionare su tale conoscenza.

Le ontologie affondano le loro radici nella filosofia greca, in particolare nella metafisica. Nella sua accezione filosofica, una ontologia è la rappresentazione di tutto quello che esiste, dell’essere <INSERIRE RIFERIMENTO BIBLIOGRAFICO>.

In informatica esse vengono rappresentate attraverso alcuni concetti chiave: *classi*, *relazioni*, *proprietà*. Cercando di fare un po’ d’ordine diciamo che una ontologia è divisa in due parti: *T-Box* e *A-Box*.

**T-Box:** La T-Box è la definizione terminologica dell’ontologia. Dei termini e delle relazioni che intercorrono tra loro. I termini vengono definiti usando il concetto di classe e proprietà. Una classe, così come una proprietà, viene identificata da un URI (Universal Resource Identifier). <DEFINISCI QUI UN PICCOLO ESEMPIO DI TBOX: 2-3 CLASSI E E UN PAIO DI RELAZIONI. LO STESSO ESEMPIO LO RIPRENDI DOPO PER LA ABOX> Le proprietà possono essere di due tipi: *data type property* e *object property*.

**A-Box:** La A-Box contiene le asserzioni dell’Ontologia. Ovvero l’insieme degli individui, o istanze delle classi definite nella T-Box. Gli individui sono la parte estrema dell’Ontologia e rappresentano i dati concreti. <FAI UN ESEMPIO CHE SIA COERENTE CON L’ESEMPIO FATTO PER LA TBOX> Ad esempio se nella T-Box è definita la classe *Impiegato*, un individuo istanza di questa classe sarà formato partendo dai dati reali di un impiegato (es. Nome: Mario, Cognome: Rossi ecc). La creazione della A-box permette al tempo stesso di convalidare con esempi pratici la qualità della terminologia creata nella T-Box [1]. Il "significato" dell'individuo, nel sistema formale, risiede esattamente nelle sue relazioni con altri elementi del sistema, a partire dalle classi a cui gli individui appartengono, fino alle relazioni con altri individui stabiliti mediante l'uso di proprietà. Inoltre in contrapposizione a quello che avviene in una base di dati, tutto quello che non è contenuto nella ABox non è ne vero ne falso. Smplicemente non si conosce.[4]

<FARE ED INSERIRE UNA FIGURA CON GLI ESEMPI DI TBOX E ABOX PRIMA DEFINITI>

La maggior parte delle applicazioni che lavorano con ontologie sono cosiddette di dominio. Alcuni enti stanno lavorando alla definizione di un'ontologia superiore che possa essere adottata come standard di riferimento per lo sviluppo di ontologie di dominio. Un esempio è rappresentato da **SUMO** (<http://www.adampease.org/OP/>), gestita dall’IEEE (https://www.ieee.org). Questa vuole diventare un'ontologia superiore standard, utilizzabile da chiunque e da qualunque applicazione.

Per sviluppare un ontologia possiamo utilizzare dei linguaggi. L’utilizzo di alcuni linguaggi e dei vocabolari in essi definiti, ci permette di modellare classi e proprietà con alcuni standard riconosciuti a livello internazionale. Due esempi di questi linguaggi sono RDF e OWL. Nei paragrafi successivi verranno analizzati in dettaglio.

3.2 Perché un'ontologia

<QUI POSSIAMO DIRE CHE UN’ONTOLOGIA PUÒ ESSERE USATA PER:

DEFINIRE UNA TERMINOLOGIA COMUNE

CONSENTIRE RAGIONAMENTO AUTOMATICO

REALIZZARE MOTORI DI RICERCA

ALLINEARE/RICONCILIARE DATI ETEROGENEI, TRAMITE LA DEFINIZIONE DI REGOLE DI TRASFORMAZIONE>

3.3 Il linguaggio RDF

Uno degli obiettivi del web semantico, e in particolare delle ontologie è quello di rendere le risorse comprensibili ad agenti software. Per ottenere questo, ci si avvale del concetto di metadato, che svolge in questo campo un ruolo fondamentale. Ovvero delle informazioni che vengono aggiunte ai dati. A questo scopo il W3C propone e gestisce RDF (Resource Description Framework) [3], un linguaggio che permette di annotare una risorsa, attraverso informazioni che descrivono alcune sue caratteristiche.

L’utilizzo di metadati ci permette di dare una sorta di descrizione semantica. Questo permette di costruire motori di ricerca molto più potenti e raffinati. Alcune query che riguardano la correlazione di più dati, che oggi sono improponibili in un motore di ricerca, un domani con l’aggiunta della semantica ai dati, saranno possibili. Ad esempio il filtraggio in base alle caratteristiche di un utente.

3.3.1 Dettaglio del linguaggio RDF

La specifica RDF è costituita da due componenti: RDF Model and Syntax e RDF Schema [3]. La prima componente riguarda la descrizione del data model RDF, ovvero il modello dei dati. Per fare questo si utilizza la sintassi XML. RDF Schema invece ci permette di descrivere le proprietà e le relazioni che intercorrono tra le risorse descritte nel modello dei dati.

Ogni risorsa viene identificata univocamente da un URI (Uniform Resource Identifier).

Una risorsa può essere sia interna al web (una pagina HTML, un documento XML ecc), oppure esterna al Web. Può essere ad esempio un libro, un impiegato, un paziente ecc. Una proprietà può essere un attributo o una relazione che ci permette di descrivere la risorsa stessa. Il significato e le caratteristiche della proprietà vengono definite tramite RDF Schema. Infine è possibile definire uno statement (espressione), ovvero l’insieme dell’elemento che descrive la risorsa, di un predicato (proprietà) e di un oggetto che definisce un valore.

Lo statement tipo in. RDF è: <soggetto> HAS <predicato> <oggetto>. In questo modo possiamo descrivere una risorsa.

A close up of a logo

Description automatically generated

*Figura che ci illustra una risorsa un predicato e un valore.*

Nella figura vediamo come la risorsa Document 1, viene messa in relazione con John Smith, dal predicato Author. Vediamo il caso di una pagina web:

A screenshot of a cell phone

Description automatically generated

Abbiamo anche il caso in cui l’oggetto di uno statement è a sua volta una risorsa:

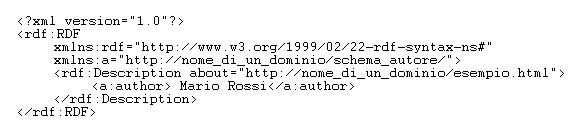
A screenshot of a cell phone

Description automatically generated

In questo modo possiamo descrivere ancora più nel dettaglio una risorsa.

3.3.2 RDF e XML

RDF prevede l'utilizzo di una sintassi basata su XML per specificare in maniera formale gli statement RDF.



In questo caso abbiamo in maniera formale (XML), l’esempio dell’autore di un sito, visto nel paragrafo precedente. L'elemento <rdf:RDF> racchiude la definizione dello statement RDF. Al suo interno abbiamo la definizione di due Namespace: il primo è il Namespace RDF, mentre il secondo Namespace contiene l'URI che identifica lo schema RDF. Utilizzato per descrivere la semantica e le convenzioni nel l'utilizzo delle proprietà presenti nello statement.

<rdf:Description> con l’attributo about identifica la risorsa. Della quale stiamo definendo il metadato.

Si può specificare in maniera più dettagliata le caratteristiche dell’autore, ad esempio aggiungendo telefono,email ecc.

A close up of a logo

Description automatically generated

Possiamo definire anche dei gruppi di risorse all’interno di un contenitore. Abbiamo 3 tipi di contenitori:

* Bag: un insieme con la possibilità di ripetizioni. In questo caso l’ordine non è rilevante.
* Sequence: è un contenitore dove è importante l’ordine. In questo caso sono ammesse le ripetizioni.
* Alternative: E’ un insieme dove è possibile selezionare un solo elemento. L’ordine è rilevante per dare una preferenza.

3.3.3 La reificazione in RDF

Con il meccanismo della reificazione è possibile definire meta-informazioni su delle meta-informazioni. A volte abbiamo la necessità di definire una meta-informazione come una risorsa da descrivere.

A questo punto avendo astratto una meta-informazione è possibile definire delle proprietà su di essa.

A screenshot of a cell phone

Description automatically generated

In questo caso avendo astratto la proprietà member, possiamo definire alacune caratteristiche. Nel nostro caso il ruolo in una relazione di membership[5].

3.4 Il linguaggio OWL

Sarebbe difficile oggigiorno costruire un agente software in grado di rispodere ad alcune query. Come quella data come esempio nella guida OWL rilasciata deal w3c.org[6]:

*"Dimmi quali vini dovrei comprare da servire con ogni portata del seguente menù. E comunque ricorda che non mi piace il Sauternes."*

Come nel caso del RDF anche nel caso di OWL abbiamo la possibilità di aggiungere informazioni alle singole parole di una frase. In questo caso di una query. Più nello specifico OWL ci permette di assolvere a questo compio attraverso la definizione di una Ontologia.

L’Ontologia è la scienza della descrizione del tipo di entità del mondo e di come sono correlate tra loro[6]. Una ontologia OWL può includere descrizioni delle classi, proprietà e loro istanze. Lo strato di semantica formale messo a disposizione da OWL ci permette di dare una conseguenza logica alla descrizione formale di classi, proprietà e istanze. Così da poter estrarre fatti che non sono esplicitamente descritti nell’ontologia stessa. In questo caso parleremo di derivazione logica.

Le derivazioni logiche possono essere basate su documenti distribuiti combinati tra di loro usando meccanismi OWL.

A differenza di XML Schema una ontologia non è solamente la definizione dei messaggi. Ma una vera e propria rappresentazione della conoscenza. Attualmente la maggior parte degli standard Web definisce solo il formato dei messaggi e degli standard di comunicazione. Questo ci permette di dare una semantica ad un messaggio solamente al ricevimento del medesimo, una semantica valida solo in quel contesto.

Un vantaggio delle ontologie è quello di avere strumenti in grado di ragionare su di esse. Indipendentemente da un particolare contesto, dominio[6].

. E’ un componente dell'attività del W3C per il Semantic Web. Lo scopo è quello di rendere le risorse sul Web più prontamente accessibili ai processi automatizzati. A questo fine vengono aggiunte informazioni riguardanti le risorse che descrivono e forniscono i contenuti del Web. Poiché il web semantico ha una natura distribuita OWL deve permettere la raccolta di informazioni da fonti distribuite. OWL permette una soluzione parziale a questo problema, in quanto è possibile correlare le ontologie tra loro e importare una intera ontologia all’interno di un’altra[6].

Del linguaggio OWL esistono tre sotto-linguaggi: OWL Lite, OWL DL e OWL Full.

OWL Lite: è utilizzato quando abbiamo bisogno di definire semplicemente di una gerarchia di classificazione e semplici restrizioni su di essa. Esso permette valori di cardinalità pari a 0 o 1.

OWL DL: permette la massima espressività, senza perdere la completezza computazionale. Ovvero tutte le conclusioni hanno la garanzia di essere calcolate in un tempo definito. Abbiamo in questo modo la decidibilità dei sistemi di ragionamento. OWL DL comprende tutti i costrutti del linguaggio OWL, con alcune restrizioni. Tra le quali citiamo quella della separazione del tipo: una classe non può essere un individuo o una proprietà, così come una proprietà non può essere un individuo o una classe. DL sta per Description Logic. Frammento decidibile della logica del primo ordine.

OWL Full: in questo caso abbiamo la massima espressività messa a disposizione da RDF. Il limite è che non abbiamo garanzie computazionali. Ovvero il calcolo di una conclusione potrebbe anche non terminare. Ad esempio in OWL Full una classe può essere trattata contemporaneamente come una collezione di individui e come un individuo a pieno titolo. OWL Full permette di aumentare il significato di un vocabolario predefinito in RDF o OWL. E’ improbabile che un software di ragionamento sia in grado di sostenere un ragionamento su tutte le caratteristiche di OWL Full.

Ciascuno di questi sotto-linguaggi è un’estensione del modello più semplice di OWL. Sia in ciò che può essere espresso sia in ciò che può essere concluso.

La scelta di un linguaggio piuttosto che un altro dipende dalle necessità del dominio in cui si intende impiegarlo. E’ possibile passare da un documento RDF ad una ontologia OWL DL o OWL Lite. Facendo attenzione alle limitazioni imposte da questi linguaggi. Il passaggio da RDF a OWL Full è più semplice visto che in questo caso abbiamo a disposizione l’intera espressività RDF.

Inoltre possiamo dire:

* Ogni ontologia OWL Lite legale è un'ontologia OWL DL legale.
* Ogni ontologia OWL DL legale è un'ontologia OWL Full legale.
* Ogni conclusione OWL Lite valida è una conclusione OWL DL valida.
* Ogni conclusione OWL DL è una conclusione OWL Full valida.

OWL addotta una assunzione di mondo aperto. Sebbene una classe viene definita in una ontologia O1, tale definizione può essere estesa in altre ontologie. Con la restrizione che le nuove informazioni della classe non possono essere in contraddizione.

OWL può essere considerato un estensione semantica del vocabolario RDF.

3.4.1 Le classi OWL

Abbiamo bisogno di una sintassi e di una semantica per il linguaggio OWL. Per definirle un ruolo molto importante viene svolto dallo spazio dei nomi. Attraverso il quale possiamo definire un insieme di termini da esso individuato.

<rdf:RDF xmlns:owl ="http://www.w3.org/2002/07/owl#"  
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
 xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">

Questa è una definizione base dello spazio dei nomi per un documento OWL. Lo spazio dei nomi viene racchiuso nel tag *<rdf:RDF*. Come si può vedere viene importato lo spazio dei nomi *XMLSchema,* per poter utilizzare il vocabolario XML Schema. Gli elementi del vocabolario XML Schema sono accessibili tramite il prefisso *xmls:*, definiti all’indirizzo *http://www.w3.org/2001/XMLSchema#*. Inoltre viene dichiarato lo spazio dei nomi OWL, questo significa che gli elementi che presentano il prefisso *owl:,* devono essere interpretati partendo dallo spazio dei nomi definito all’indirizzo *<http://www.w3.org/2002/07/owl#>*. Viene dichiarato anche lo spazio dei nomi rdf, un discorso analogo quindi può essere fatto per gli elementi che presentano il prefisso *rdf:* riconosciuti come quelli definiti all’indirizzo *<http://www.w3.org/1999/02/22-rdf-syntax-ns#>*. Per quanto riguarda gli elementi che presentano il prefisso *rdfs:* essi individuano oggetti che derivano dallo spazio dei nomi *rdf-schema*.

Stessa cosa può essere fatta per poter utilizzare lo spazio dei nomi definito da altre ontologie.

La maggior parte di una ontologia OWL è composta da classi proprietà, istanze di classi e le relazioni tra queste istanze. La struttura di una ontologia OWL è un albero formato dalle classi e le proprietà, ogni classe dell’albero eredita dalla classe owl:Thing. OWL permette anche la definizione della classe vuota owl:Nothing. Un esempio di dichiarazione di una classe in OWL è:

<owl:Class rdf:ID="Book"/>

All’interno dell’ontologia che stiamo scrivendo possiamo riferirci direttamente con #Book ad esempio rdf:resource="#Book" mentre in altre ontologie dobbiamo riferirci con il nome completo, ad esempio:

"http://www.w3.org/TR/2004/Mia-ontologia-owl/library#Book"

Un altro modo per riferirsi ad una classe è attraverso il costrutto rdf:about="#Book", in questo caso possiamo estendere la definizione di una risorsa. In una ontologia possiamo estendere la definizione di una classe definita in un’altra ontologia. Senza modificare il contenuto del documento originale dell’ontologia. Il costrutto che ci permette di realizzare questo è:

rdf:about="&ont;#x"

Con &ont; interpretabile come l’indirizzo del documento originale dell’ontologia a cui appartiene la classe che intendiamo estendere.

Il costrutto fondamentale per costruire una tassonomia di classi è rdfs:subClassOf. Esso permette di mettere in relazione una classe con una classe più generica. Se una classe X è una sottoclasse di Y, allora ogni sottoclasse di X sarà una sottoclasse di Y. Ad esempio:

<owl:Class rdf:ID="YellowBook">

<rdfs:subClassOf rdf:resource="#Book" /> ...

</owl:Class>

Estendendo quanto detto finora, potremmo pensare ad una ontologia che contiene definizioni di classi generiche per un particolare dominio. Importando questa ontologia potremo specificare le sue classi per un particolare contesto.

3.4.2 Gli individui OWL

Un individuo è definito partendo dalla sua appartenenza ad una classe, ad esempio:

<AdventureBook rdf:ID="MobyDick" />

Un punto importante da notare è che la definizione di un individuo e la definizione della classe a cui esso appartiene potrebbero essere anche su file differenti, quindi su ontologie differenti. Tutto questo è reso possibile dal meccanismo dell’importazione di una ontologia all’interno di un’altra ontologia. Sull’importazione è importante dire che è molto utile importare ontologie ampiamente condivise, questo ci garantisce che anche quella che stiamo costruendo lo sarà. E’ comunque molto arduo combinare tra loro diverse ontologie, e gli strumenti che utilizziamo ci dovrebbero fornire supporto per questo fine. Torneremo successivamente su questo punto.

Capire se un elemento può essere o meno trattato come un individuo non è banale. Un elemento può essere considerato come un individuo quando non sono possibili ulteriori specificazioni. Mentre una classe è semplicemente un nome e una collezione di proprietà che descrivono un insieme di individui. Occorre anche dire che una classe può essere considerata sia come classe che come individuo a seconda del contesto in cui ci troviamo ad operare. Ad esempio la malattia di Alzheimer può essere considerata come un individuo se ci troviamo a specificare una ontologia generica delle malattie, mentre può assumere addirittura il ruolo di classe radice se ci troviamo a dover specificare proprio l’ontologia di questa malattia.

In OWL Full è possibile trattare una istanza di una classe come una classe che ha a sua volta delle istanze. Restando sull’esempio della malattia di Alzheimer, la malattia di Alzheimer da considerarsi come istanza in una ontologia delle malattie, potrebbe a sua volta avere altre istanze andando a specificare le singole malattie dei singoli pazienti [6].

In conclusione possiamo dire che lo sviluppo di una ontologia deve essere fortemente guidato dall’uso che se ne intende fare.

3.4.3 Le proprietà in OWL

Oltre alle classi e agli individui in OWL è possibile definire delle proprietà. Che definiscono fatti generali per le classi e fatti specifici per gli individui.

Una proprietà è una relazione binaria. Possono essere definite due tipi di proprietà:

**Data type properties**: relazioni tra istanze di classi e elementi *RDF literals* o *XML Schema*.

**Object properties**: relazioni tra istanze di due classi.

Distinguiamo le proprietà quindi, in due categorie. Se definiscono un legame tra un individuo e una classe la chiameremo Object property. Se definiscono invece un legame tra una proprietà e un tipo di dato, ci troveremo di fronte ad una Data type property. Le proprietà sui tipi di dati possono essere definite in accordo ai litterals RDF oppure come semplici XML Schema datatypes.

OWL utilizza la maggior parte degli XML Schema datatypes. I loro riferimenti vengono costruiti partendo dall’URI base: <http://www.w3.org/2001/XMLSchema>. Che insieme ai rdfs:Literal formano i tipi di dati utilizzati in OWL. Tutti i reasoner che vengono applicati su documenti OWL hanno come requisito fondamentale quello di supportare i tipi xsd:integer e xsd:string.

Ad una proprietà possono essere associate delle restrizioni, ad esempio può essere associato un range (intervallo) al quale essa si applica, che rappresenta il punto di partenza della proprietà. Il dominio di una proprietà invece rappresenta il punto di arrivo. Ovvero solo dalle classi di un intervallo può partire una proprietà e solo nelle classi del dominio può atterrare una proprietà.

<owl:ObjectProperty rdf:ID="writeBy"> ...

<rdfs:domain rdf:resource="#Writer" />

<rdfs:range rdf:resource="#Book" /> </owl:ObjectProperty>

In questo modo nella nostra ontologia abbiamo definito un intervallo e un dominio per la proprietà #writeBy. Stiamo dicendo che la proprietà #writeBy può atterrare solamente nell’intervallo degli scrittori (#Writer), e può essere associata solo ad uno libro (#Book). Tale scelta è ovviamente opinabile, ma potremmo definire questa restrizione se utile per gli scopi della nostra ontologia.

Potremo definire anche altre restrizioni, ad esempio potremo dire che un oggetto #Book ha almeno uno scrittore (#Writer) che l’ha scritto:

<owl:Restriction>  
 <owl:onProperty rdf:resource="#writeBy"/>  
 <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>

</owl:Restriction>

Per una proprietà è possibile anche definire domini multipli, in questo caso il dominio della proprietà è identificato dall’intersezione di questi domini. Stesso discorso può essere fatto per l’intervallo [6]. Da notare che l’uso degli intervalli e dei domini è differente dall’uso dei tipi in un linguaggio di programmazione classico. In OWL un intervallo può essere utilizzato per definire il tipo di un dato. Inoltre è possibile definire ulteriori restrizioni per particolari sottoclassi di una classe. Con i meccanismi owl:someValuesFrom e owl:allValuesFrom. Ad esempio potremo decidere che un libro di matematica ha un autore che è un matematico applicando owl:allValueFrom. Mentre applicando owl:someValuesFrom potremo dire che se un libro di matematica ha più autori almeno uno di essi è un matematico.

In OWL una proprietà può essere:

**Transitiva**: se una proprietà viene definita transitiva con il costrutto <rdf:type rdf:resource="&owl;TransitiveProperty" /> significa che se la proprieta è valida tra X e Y e tra Y e z allora è valida anche tra X e Z. Ovvero P(x,y) e P(y,z) implicano P(x,z).

**Simmetrica**: se una proprietà viene definita simmetrica con il costrutto <rdf:type rdf:resource="&owl;SymmetricProperty" /> significa che se la proprietà e valida tra X e Y allora è valida anche tra Y e X. Ovvero P(x,y) se e solo se P(y,x).

**Funzionale**: se una proprietà è definita come funzionale con il cstrutto <rdf:type rdf:resource="&owl;FunctionalProperty" /> allora se la proprietà e valida tra X e Y e tra X e Z questo signigica che Y e Z sono uguali. Ovvero P(x,y) e P(x,z) implicano che y = z.

**Inversa**: se una proprietà viene definita inversa con il costrutto <rdf:type rdf:resource="&owl;InverseFunctionalProperty" /> significa che se una proprietà è valida tra Y e X e tra Z e X questo significa che Y e Z sono uguali. Ovvero P(y,x) e P(z,x) implica y = z.

Possono essere definite delle sotto-proprietà che consistono in una ulteriore specificazione. Infatti come per le classi anche per le proprietà è possibile definire una gerarchia.

3.4.4 Combinazione di più ontologie in OWL

Come abbiamo accennato in precedenza con il meccanismo dell’importazione di una ontologia all’interno di un’altra è possibile combinare il contenuto informativo di più ontologie in maniera da rendere più completa una terza che stiamo costruendo. A questo scopo sono molti importanti i supporti che ci offre lo strumento che abbiamo scelto per l’implementazione.

I costrutti OWL equivalentClass e equivalentProperty ci permettono di stabilire equivalenze tra proprietà e classi provenienti da diverse ontologie. Questo ci permette di fare ordine nella nostra implementazione.

Un rischio nel combinare più ontologie tra di loro è quello di trovarci di fronte a delle contraddizioni, ci può essere una ontologia che ci dice che tutte le A sono B e una’altra che invece afferma che tutte le A NON sono B. In questo caso ad esempio non potremo definire nessuna relazione e nessun individuo che soddisferà la combianzione delle ontologie.

Con la proprietà equivalentClass affermeremo che due classi hanno precisamente lo stesso insieme di istanze. Ad sempio:

<owl:Class rdf:ID="Book">  
<owl:equivalentClass rdf:resource="&otherLibraryOntology;Book"/>

</owl:Class>

Questa proprietà è molto utile quando abbiamo ad esempio due ontologie che utilizzano due nomi diversi per definire la stessa classe.

Con il costrutto sameAs invece possiamo definire l’equivalenza tra individui. Il meccanismo è molto simile a quello utilizzato per stabilire l’equivalenza tra classi.

<Book rdf:ID="FedericoFavoriteBook">  
<owl:sameAs rdf:resource="#MobyDick" />

</Book>

In questo caso potremo riferirci al libro Moby dick sia come l’individuo #MobyDick definito in una ontologia sia come l’individuo libro preferito di Federico (FedericoFavoriteBook) in un’altra ontologia. E da notare che in OWL non viene fatta un assunzione di nome unico per un individuo. Questo significa che potremmo avere lo stesso individuo chiamato con due nomi differenti. La proprietà sameAs può essere anche utilizzata per uguagliare una classe ad un individuo, una classe ad una proprietà etc. Facendo in modo che ogni lato dell’associazione venga trattato come individuo.

Eistono anche delle proprietà che ci permettono di definire le differenze tra individui. I costrutti differentFrom, AllDifferent.

<BookType rdf:ID="YellowBook" />

<BookType rdf:ID="YellowBook"> <owl:differentFrom rdf:resource="#MathematicBook"/>

</BookType>

Questo costrutto è utile quando abbiamo una proprietà, ad esempio hasType ad un libro, e imponiamo che questa proprietà può assumere solamente un valore. Definendo un libro sia come un libro giallo che come un di matematica potremo asserire che genere giallo e genere matematica sono la stessa cosa. Invece avendo definito la proprietà differentFrom, come sopra, otteniamo una contraddizione.

Esiste un modo più semplice per definire la differenza tra individui:

<owl:AllDifferent>  
<owl:distinctMembers rdf:parseType="Collection">

<lib:BookType rdf:about="#Yellow" />

<lib:BookType rdf:about="#Mathematic" />

<lib:BookType rdf:about="#Adventure" />

</owl:distinctMembers>

</owl:AllDifferent>

In questo modo definiamo come differenti tutti i valori di BookType all’interno della collection.

Esitono ancora altri costrutti OWL ma non è lo scopo di questa tesi approfondire ulteriormente le caratteristiche del linguaggio OWL. Per questo fermiamo la nostra descrizione a questo punto.

3.5 Strumenti per l'editing: Protegé

Protegé è uno strumento per l’editing di ontologie sviluppato dall’università di Stanford in California. Costituisce l’alternativa più completa e diffusa per l'implementazione di ontologie. Ed è lo strumento utilizzato per lo sviluppo di questa tesi.

Iniziamo la nostra analisi di Protegé andando a vedere cosa ci viene proposto nella schermata iniziale:

A screenshot of a social media post

Description automatically generated

Come si può vedere la prima cosa che ci viene proposta è una lista di plug-in che possiamo installare e in questo modo ampliare le funzionalità di Protegé.

A screenshot of a social media post

Description automatically generated

Chiudendo la finestra per l’installazione dei plug-in la prima cosa che notiamo è una serie di tab, che ci permette di cambiare vista e operare sui vari set di funzionalità, ognuno dei quali si riferisce ad uno specifico aspetto di una ontologia:

**Active ontology**: ci permette di gestire le informazioni generali dell’ontologia.

**Entities**: è forse il tab più importante di Protegé. E apre una serie di sotto-tab:

A screenshot of a cell phone

Description automatically generated

In questa vista possiamo gestire l’implementazione delle classi, delle proprietà (Object property, Data type property), le annotation property, i datatype e gli individui. Torneremo sulla descrizione più dettagliata di questi tab.

**Individuals by class**: in questo tab possiamo avere una visuale degli individui divisi per classe:

A screenshot of a social media post

Description automatically generated

A destra abbiamo la lista delle classi. Selezionando una classe otteniamo sotto la lista degli individui. E successivamente selezionando un individuo a destra possiamo avere le sue caratteristiche.

**DL query**: in questo tab possiamo eseguire delle query, che ci permettono di estrarre dati dalla nostra ontologia:

A screenshot of a cell phone

Description automatically generated

Ora andiamo ad analizzare il tab entities. Che come detto in precedenza rappresenta il tab con più funzionalità e quello che ci permette di costruire la base della nostra ontologia. Per fare questo andiamo ad analizzare i sotto-tab di questa vista.

Il primo tab che andiamo ad analizzare è il tab classes:

A screenshot of a cell phone

Description automatically generated

In questo tab possiamo sia visualizzare le caratteristiche delle classi che abbiamo inserito, sia aggiungerle, cancellarle o modificarle.

Nel tab Object properties:

A screenshot of a cell phone

Description automatically generated

Possiamo visualizzare, inserire, modificare e cancellare le object properties della nostra ontologia.

Stesso discorso possiamo fare per il tab data type properties. In questo caso possiamo visualizzare, inserire, modificare e cancellare le data type properties.

A screenshot of a cell phone

Description automatically generated

Nel tab Datatypes abbiamo una visuale dei datatype ammessi nella nostra ontologia. Da notare che i tipi di default derivano dai vocabolari OWL, RDF e XML Schema:

A screenshot of a cell phone

Description automatically generated

In questo caso possiamo solo aggiungere o cancellare un data type.

Nel tab individuals invece abbiamo una visuale degli individui presenti nella nostra ontologia, e delle sue caratteristiche:

A screenshot of a cell phone

Description automatically generated

Infine andiamo a vedere il tab Annotation properties:

A screenshot of a cell phone

Description automatically generated

In questo caso abbiamo una lista delle annotazione che possiamo associare ad una classe, proprietà, individuo o datatype.

4. SVILUPPO DELL'ONTOLOGIA RELATIVA AI TEST NEUROPSICOLOGICI

Il materiale da cui siamo partiti, nello sviluppo della nostra ontologia, sono i risultati dei test del data base ADNI. Questi dati vengono distribuiti su dei file csv come raw data. Il loro utilizzo quindi risulta abbastanza difficile, quando si vogliono comprendere le relazioni che ci sono fra i vari dati.

Il nostro obbiettivo è stato quello di fornire un’organizzazione in modo tale da poter estrarre informazioni dai dati nel loro insieme. Per fare questo abbiamo scelto di organizzarli in una ontologia. Abbiamo sviluppato dapprima la T-Box, così da avere una idea terminologica del contesto. Successivamente ci siamo occupati di inserire gli individui. Per fare questa ultima operazione abbiamo utilizzato delle tecnologie (Jena e TDB) che ci permettessero da un lato di automatizzare il processo di inserimento, e dall’altro di sostenere e rendere fruibile la memorizzazione di una quantità rilevante di dati. Non sarebbe stato possibile infatti interrogare i dati utilizzando Protegé, che è stato invece lo strumento utilizzato per sviluppare la T-Box.

Per interrogare i dati abbiamo utilizzato SPARQL. Questo è stato possibile realizzando un client che ci consentisse, inserendo una query SPARQL, di estrarre le relative triple nel Data Base TDB.

Nei prossimi paragrafi verranno descritte le varie fasi dello sviluppo di questo progetto.

4.1 Approccio e metodologia nel trattamento dei dati

Siamo partiti dall’analisi dei dati distribuiti da ADNI. In particolare ADNI distribuisce i dati in dei file csv, come raw data. Ogni test quindi ha un file csv dove vengono inseriti i risultati di ogni paziente. Il csv è composto da una serie di campi. I campi sono sia le prove, di cui il test è composto, sia alcune informazioni accessorie, come l’identificativo numerico del paziente,

Da questa analisi abbiamo prodotto dei fogli Excel. In particolare per ogni tipo di test, preso in considerazione, abbiamo prodotto un foglio Excel, nel quale sono state inserite le informazioni relative ad ogni singolo campo del csv, relativo al test, distribuito da ADNI. Nel fare questo siamo stati aiutati da un particolare documento, a sua volta un file csv, fornito da ADNI, chiamato Data Dictionary. Contenente le informazioni relative ai campi di tutti i csv distribuiti da ADNI.

Le informazioni che abbiamo considerato e riportato nei nostri documenti sono:

* **Description**: una breve descrizione del significato del campo.
* **ADNI tag meaning**: la descrizione del tag relativo al campo.
* **Name of the test**: il nome del test a cui il campo si riferisce.
* **ADNI1,ADNIGO,ADNI2,ADNI3**: una serie di 4 campi contenente l’informazione relativa alla presenza o meno del campo ADNI nelle varie fasi.
* **Scoring**: una descrizione di come viene assegnato lo score alla particolare prova del test a cui il campo ADNI fa riferimento.
* **Domain**: il dominio della particolare prova del test a cui il campo ADNI fa riferimento.
* **Subdomain**: il sotto-dominio della particolare fase del test a cui il campo ADNI fa riferimento.
* **Tag**: il nome del campo nel csv fornito da ADNI. Csv contente i risultati del test che stiamo considerando.

Come abbiamo accennato in precedenza, questo tipo di lavoro è stato svolto per ogni csv relativo ai test che abbiamo considerato.

Da questa analisi preliminare siamo partiti per costruire la parte terminologica (T-Box), della nostra ontologia. Le descrizioni, ad esempio, sono state inserite come annotazioni negli elementi (classi) della nostra ontologia. Il dominio, trasversale ai vari csv forniti da ADNI, è stato utilizzato per costruire la tassonomia delle classi. Il campo Scoring è stato molto utile per le Data type property, delle prove dei test, che hanno uno score. Torneremo nei successivi paragrafi, più nel dettaglio e con degli esempi, sull’utilizzo che abbiamo fatto di questi fogli Excel.

Come abbiamo già detto in precedenza, nel nostro lavoro ci siamo concentrati su sei test (FAQ, MMSE, CDR, GDS, NPI e NPI-Q), che abbiamo ritenuto maggiormente significativi. Ognuno di questi test è fatto da una serie di domande, esercizi pratici ecc. Gli score o più in generale i risultati delle prove non vengono censiti nello stesso modo. Alcune prove hanno una scala che va da uno a tre, altre da uno a cinque e così via, altre prove consistono in delle domande che possono avere risposta Yes/No. Inoltre non sempre queste prove sono presenti in tutte le fasi di ADNI. Alcune domande presenti in ADNI-1, ad esempio, potrebbero non essere presenti in ADNI-3, e viceversa. Questi fattori rappresentano delle difficoltà nel trattamento dei dati. Per quanto riguarda le scale abbiamo provveduto a creare le corrispettive Data type property nella nostra ontologia. Considerando anche che in questo caso le scale possono cambiare anche nello stesso test, a seconda della prova.

4.2 Sviluppo della TBox, Terminological Box

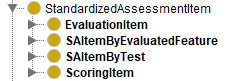
Nell’implementazione della T-Box abbiamo creato alcuni termini radice. Uno di questi è il concetto di evento. Ogni test, nella nostra ontologia, rappresenta un evento. I test nel loro insieme sono stati censiti sotto la classe *StandardizedAssessment*, che a sua volta è sotto-classe di evento. Le singole prove dei test invece sono state raccolte nella parte bassa dell’ontologia, sotto la classe *StandardizedAssessmentItem*. Cercheremo di riferirci alle prove dei test con il termine di **item,** nel prosieguo di questa tesi.

Nel censire gli item abbiamo cercato di individuare delle caratteristiche trasversali ai vari test. Ovvero abbiamo creato delle classi, su cui poi far confluire gli item dei vari test, che rispondessero a delle precise caratteristiche. Ad esempio sotto la classe *EatingItem* sono andati a finire in gran parte item del test NPI (Neuropsychiatric Inventory), due item riguardanti il test FAQ (Functional Assessment Questionnaire) e un item riguardante il test NPI-Q (Neuropsychiatric Inventory Questionnaire).

A close up of text on a white background

Description automatically generated

Gli *StandardizedAssessmentItem* sono stati divisi in 4 classi: *EvaluationItem*, *SAItemByEvaluatedFeature*, *SAItemByTest*, *ScoringItem*.



Questa suddivisione intende rispecchiare le varie tipologie di test che vengono eseguite su un paziente. Sotto la classe EvaluatedItem abbiamo inserito tutti quei test che hanno una risposta aperta. Domande del tipo: *Are you basically satisfied with your life?* oppure *What season is it?*.

Sotto la classe *SAItemByEvaluatedFeature*, sono stati inseriti tutti quegli Item che concentrano la valutazione su una particolare caratteristica, ad esempio la preparazione di un pasto bilanciato.

Sotto la classe *SAItemByTest* sono stati inseriti tutti quegli item che hanno un test come oggetto di una valutazione, ad esempio la ripetizione di una parola al contrario.

Sotto la classe *ScoringItem* sono stati inseriti quegli item che hanno un preciso score, su una scala definita, come risultato della valutazione.

Da notare che una classe può essere sotto-classe di una o più di queste classi.

4.3 Inserimento degli individui la A-Box, Assertion Box

Una delle funzioni della costruzione di una A-Box è quella di validare la consistenza della T-Box. Anche nel nostro caso questa regola è stata rispettata. Infatti è proprio grazie all’inserimento degli individui che ci siamo resi conto di alcuni buchi e mancanze della nostra T-Box, ad esempio alcuni item che non avevavamo inserito.

Per l’inserimento degli individui nella A-Box abbiamo fatto uso delle API Jena e di un data base TDB. Il Data base memorizza i dati in forma di triple, per questo viene chiamato triple store. Il concetto di tripla ha un ruolo fondamentale, nella memorizzazione dei dati all’interno di una ontologia. In un triple store si ragiona in termini di triple, si memorizzano e si estraggono triple.

Nel nostro lavoro siamo partiti dai csv distribuiti da ADNI. Abbiamo implementato un motore che legge questi file csv e li memorizza in delle strutture dati Java. Successivamente queste strutture dati vengono passati al modulo che li memorizza nel nostro triple store. Quest’ultimo modulo per svolgere il proprio compito fa uso delle API Jena. A monte di queste due operazioni c’è l’inizializzazione del Data Base, che viene eseguita partendo dal file OWL che abbiamo esportato da Protegé, contenente la T-Box della nostra ontologia.

A screenshot of a cell phone

Description automatically generated

Lo step preliminare, vale a dire il caricamento della T-Box, viene eseguto ogni qual volta vi appartiamo una modifica. Modifica eseguita utilizzando Protegé.

Una volta caricata la T-Box, e caricati gli individui possiamo eseguire le query SPARQL con il nostro client.

[1] Modellazione e editing di ontologie - Rappresentazione e algoritmi - Laurea magistrale in Cinema e Media Modulo II Dispensa II

[2] The Alzheimer’s Disease Neuroimaging Initiative Informatics Core: A Decade in Review Arthur W. Toga and Karen L. Crawford Laboratory of Neuro Imaging (LONI), Institute for Neuroimaging and Informatics (INI), University of Southern California, Los Angeles, CA 90033

[3] Introduzione a RDF (Resource Description Framework) <http://www.ce.unipr.it/people/bianchi/Teaching/>

[4] Web Semantico da un sogno di Tim Berners-Lee Silvio Peroni

[5] Reificazione (informatica) <https://it.wikipedia.org/wiki/Reificazione_(informatica)>

[6] OWL Linguaggio Ontologico per il Web Guida www.w3c.org

1. https://www.efpia.eu/ [↑](#footnote-ref-1)
2. https://www.imi.europa.eu/ [↑](#footnote-ref-2)
3. https://www.synapse.org/#!Synapse:syn2790911/wiki/235388 [↑](#footnote-ref-3)