**SVILUPPO DI UN SISTEMA DEDUTTIVO E INTERROGAZIONE DI ONTOLOGIE NEL WEB SEMANTICO**

Il Web è diventato uno strumento ricco di contenuti e viene utilizzato ogni giorno da milioni di utenti, utilizzando un motore di ricerca per poter trovare le informazioni desiderate.

La ricerca non sempre permette di poter ottenere dei risultati soddisfacenti e in quel caso è necessario avvalersi di un raffinamento della ricerca da parte dell’utente.

Inoltre, l’incremento di applicazioni e servizi web sta conducendo alla nascita di operazioni complesse come la catalogazione, la ricerca e la composizione degli stessi.

Per favorire lo sviluppo della catalogazione e della ricerca delle risorse, come pagine Web, documenti, programmi e Web server, è opportuno introdurre una descrizione semantica di costrutti universali, in modo da favorire l’automazione di operazioni svolte finora dall’utente grazie all’interpretazione delle risorse comprensibili per il linguaggio macchina.

L’obiettivo principale che si prefigge il web semantico è quello di limitare fortemente l’intervento umano durante la ricerca d’informazioni dettagliate, attraverso la definizione di linguaggi che ne descrivono la semantica e di regole per l’interpretazione corretta. Al giorno d’oggi, infatti, la maggior parte delle risorse presenti in Internet, non è interpretabile dalle macchine, ovvero non ha un significato per i computer, ma solo per gli esseri umani. È necessario perciò l’intervento umano affinché si possa ottenere una ricerca affine per ottenere il risultato desiderato. Spesso s’inizia una ricerca con un termine generico e successivamente si affina la ricerca filtrando i risultati. Oppure bisogna accedere ad un registro per capire quale Web Service deve essere impiegato per la ricerca specifica, e solo successivamente richiamare il servizio, mentre sarebbe più consono se un agente software fosse in grado di stabilire quale servizio richiamare interpretandone la semantica.

In questo contesto interviene così il Web Semantico, definendo opportuni linguaggi per la rappresentazione delle risorse e per la loro interrogazione.

Il Web semantico rappresenta una soluzione alle limitazioni imposte dal Web moderno. Il progetto è basato su un piano molto preciso: ridefinire e ristrutturare le risorse del web in modo che il significato sia accessibile non solo da utenti umani ma anche da applicazioni software. In questo modo è possibile che le macchine possano accedere ad un insieme strutturato di informazioni e ad un insieme di regole da utilizzare per il ragionamento automatico, e quindi potranno manipolare, integrare e rendere disponibili tali risorse per applicazioni software. In questo modo le macchine potranno accedere ad un insieme strutturato d’informazioni e ad un insieme di regole da utilizzare per il ragionamento automatico, e quindi potranno manipolare, integrare e rendere disponibili tali risorse per applicazioni software oltre che per gli utenti.

I motori di ricerca utilizzano tutte le informazioni ottenute da attributi di linguaggio HTML per poter accedere con facilità alla consultazione e tutte le parole presenti nella descrizione e testo del link per poter determinare il legame tra le risorse.

Il Web semantico ha alla base tre strumenti fondamentali:

* *agenti software intelligenti*: programmi che sappiano interpretare correttamente le informazioni circa la semantica delle risorse;
* *annotazioni semantiche*: dati che descrivono le risorse facendo uso di linguaggi di rappresentazione della conoscenza;
* *ontologie*: schemi concettuali esaustivi e univocamente interpretabili che definiscono un dato dominio

La semantica è espressa da opportuni linguaggi, mentre l’interpretazione della stessa consiste nell’applicazione di una serie di regole di deduzione. Tali regole sono chiamate *regole d’inferenza* e lo strumento software che svolge questo compito è il *reasoner*, a cui è affidato il compito di derivare nuove informazioni a partire da quelle che si hanno a disposizione applicando opportune regole d’inferenza, derivate dal linguaggio con cui è espressa la base di conoscenza (*KB: Knowledge Base*) e possono essere integrate con altre.

A questo proposito, è stato preso in considerazione **LEXVO**, che deriva dal greco λεξικόν (lessico) e dal latino vocabularium (o vocabolario) ed è il nome di un progetto che mira a fornire dei servizi legati al lessico sul web. Gli Uri, considerati come ID identificatori per elementi sul web come siti Web, oggetti del mondo reale o anche entità un po' più astratte, sono il primo passo per poter fornire servizi legati al lessico sul web.

Questi utilizzano un normale browser web per poter accedere agli URI e poter ottenere le informazioni richieste in una pagina web decifrabile dall’utente.

Per poter costruire URI Lexvo si dispone di Java API che consente di creare URI per linguaggi e termini/parole, facendo riferimento alle lingue, al [link](http://www.lexvo.org/linkeddata/lexvoapi.zip) indicato da cui è possibile scaricare le risorse. La ricerca di queste risorse è stata effettuata cercando di comprendere al meglio il funzionamento di [Lexvo.org](http://www.lexvo.org/index.html) al [link](http://www.lexvo.org/linkeddata/faq.html) indicato.

Uno dei vantaggi dell’utilizzo degli URI è quello di essere gestiti dalla Library of Congress. Esiste uno [schema](http://www.lexvo.org/linkeddata/details.html) semplice ben definito per poter trasformare codici standard ISO 639-3 autorevoli in Lexvo.org e viceversa. Gli URI presenti in Lexvo.org sono stabili e non diventeranno privi di significato in futuro.

Un’altra risorsa presente è la descrizione del set di dati disponibile per essere leggibile dal linguaggio macchina a questo [link](http://www.lexvo.org/linkeddata/void.rdf) in formato VoiD.

Nella sezione presente a questo [link](http://www.lexvo.org/linkeddata/tutorial.html) è possibile leggere una sezione dedicata ai dataset e agli sviluppatori che permette di verificare e comprendere al meglio come funziona la costruzione delle URI oltre all’utilizzo delle Java API citate precedentemente. Il link dei [dettagli tecnici](http://www.lexvo.org/linkeddata/details.html) permette di poter costruire le URI in modo autonomo con le risorse disponibili.

Affinché sia possibile lavorare con i dati Lexvo, è possibile poter scaricare il [dump RDF](http://www.lexvo.org/linkeddata/resources.html) e poter utilizzare JENA o Redland per poter caricare il dump.

Una fonte ulteriormente disponibile è l’accesso al SPARQL ai dati Lexvo.org, attualmente disponibile solo tramite il server LOD <http://lod.openlinksw.com/sparql>. L’alternativa al server più veloce per poter configurare SPARQL in pochi minuti è: [RedStore](https://www.aelius.com/njh/redstore/), [Jena Fuseki](http://http/jena.apache.org/), [BlazeGraph.](https://github.com/blazegraph/database/wiki/Main_Page)

Sono presenti anche set aggiuntivi che contendono dati semantici, che includono: [Universal WordNet](http://wordnets.org/), offre la conoscenza semantica in 200lingue, [FrameBase,](http://wordnets.org/) rappresenta la conoscenza tramite frame semantici, [Etymological Wordnet](http://gerard.demelo.org/berkeley/) dati sulle origini delle parole, e [WebChild](https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/commonsense/webchild), la conoscenza del senso comune.

L’attenta analisi di Lexvo.org ha permesso di comprendere al meglio i passi per poter analizzare ed esplorare le informazioni fornite dallo stesso e iniziare a navigare per poter comprendere le relazioni che esistono tra lingue, parole, zone geografiche e così via.

Per poter iniziare ad utilizzare gli identificatori, il primo passo è comprendere dove i dati si riferiscono e a quali lingue, caratteri o entità linguistiche. Per poter determinare quali URI poter utilizzare per le entità relative alla lingua, è possibile usufruire di tre opzioni:

* per le lingue vengono fornite delle tabelle di mappatura tra gli standar ISO 639 e gli URI di Lexvo.org;
* la pagina dei [dettagli tecnici](http://www.lexvo.org/linkeddata/details.html) permette di costruire le URI
* per i programmatori è possibile utilizzare le [Java API](http://www.lexvo.org/linkeddata/lexvoapi.zip) descritte qui di seguito:

per esprimere la propria risorsa e attribuirla ad una lingua si ha triple del modulo

my:resource lvont:language lexvo:eng . my:resource lvont:language lexvo:fra .

or

my:resource dcterms:language lexvo:eng . my:resource dcterms:language lexvo:fra

L’approccio garantisce che le lingue siano identificate con URI univoci che sono dereferenziabili. La ricerca e lo studio con i dati di Lexvo, permette di poter lavorare con esso e come era stato già indicato, è possibile scaricare il [dump RDF](http://www.lexvo.org/linkeddata/resources.html) e poter utilizzare la sua libreria.

Consultando le specifiche è possibile osservare che sono presenti dei dati aggiuntivi, che includono come già analizzato:

* [Universal WordNet](http://wordnets.org/): conoscenza semantica lessicale in oltre 200 lingue
* [FrameBase](https://www.framebase.org/): rappresentare la conoscenza tramite frame semantici
* [Etymological Wordnet](http://gerard.demelo.org/berkeley/): origini delle parole
* [WebChild](https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/commonsense/webchild): conoscenza del buon senso.

Universal WordNet (UWN) è un grande grafo della conoscenza che mira a descrivere parole, entità e concetti in oltre 200 lingue diverse in una grande struttura di rete.

Le risorse possono essere sfruttate per sviluppare attraverso JVM (JAVA, Scala, ecc.) uno o più plug- in di grandi dimensioni, che forniscono i dati completi per l'uso offline.

Per poter usare la libreria è possibile dare un occhiata al codice di esempio fornito a questo [link](https://resources.mpi-inf.mpg.de/yago-naga/uwn/SimpleExample.java.txt) e indicata come segue:

package org.lexvo.uwn.examples;

import java.io.File;

import java.io.UnsupportedEncodingException; import java.nio.ByteBuffer;

import java.nio.CharBuffer;

import java.nio.charset.CharacterCodingException; import java.nio.charset.Charset;

import java.nio.charset.CharsetEncoder; import java.util.Iterator;

import org.lexvo.uwn.Entity; import org.lexvo.uwn.Statement; import org.lexvo.uwn.UWN;

/\*\*

* Simple Example of how to use the UWN API

\*

* @author Gerard de Melo

\*/

public class SimpleExample {

/\*\*

* + @param args
  + @throws Exception

\*/

public static void main(String[] args) throws Exception {

// Instantiate UWN, providing a pointer to the plugins directory. UWN uwn = new UWN(new File("plugins/"));

// Look up all meaning entity IDs for the French word "souris".

// The ISO 639-3 code for French is "fra". For English use "eng". Please refer to

// <http://www.sil.org/iso639-3/codes.asp>

// for other language codes.

System.out.println("Identifiers for Meanings of the French word 'souris'"); Iterator<Statement> it = uwn.getMeanings(Entity.createTerm("souris", "fra")); while (it.hasNext()) {

Statement meaningStatement = it.next();

Entity meaning = meaningStatement.getObject();

System.out.println(meaning + " with weight " + meaningStatement.getWeight());

}

System.out.println();

more

// Once you have a meaning entity (or any other entity), use UWN.get() to look up

// information about them.

System.out.println("All knowledge available for one of those meanings, e.g.

descriptions, superclasses, etc.");

System.out.println("(subject, predicate, object, weight)");

Entity meaning = new Entity("s/n2332156"); // this is one of the entities you get from the first iterator

Iterator<Statement> it2 = uwn.get(meaning); while (it2.hasNext())

System.out.println(it2.next());

}

}

Al seguente [link](https://github.com/dblommesteijn/uwn-api) viene presentato un Ruby wrapper per altri linguaggi di programmazione di terze parti.

L’analisi di [WebChild](https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/commonsense/webchild) è stata effettuata dall’apposito link in qui è possibile analizzare la panoramica e la possibilità di poter reperire un database.

WebChild raccoglie vaste conoscenze estratte automaticamente e disambiguate dai contenuti web. Contiene triple che collegano nomi con aggettivi tramite delle relazioni a grana fine.

[Wordnet wrappers](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-wordnet-wrappers.html) al link indicato, permette di descrivere il database pre-elaborato di WordNet che include una varietà di operazioni sul DAG di Wordnet.

L’elenco è così costituito:

* [wordnet wrappers](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-wordnet-wrappers.html)
* [part-whole](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-partwhole.html)
* [comparative](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-comparative.html)
* [property](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-property.html)
* [activity](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-activity.html)
* [spatial](http://people.mpi-inf.mpg.de/~ntandon/resources/readme-spatial.html)

La ricerca evidenzia quanto il progetto Wordnet possa fornire risorse online rispetto ad una semplice ricerca di tipo alfabetico e permette di arrivare ad un vero e proprio dizionario basato su principi della psicolinguistica.

A tal proposito emerge la sua organizzazione e quanto la disponibilità del database definisce e descrive concetti espressi da vocaboli per lo sviluppo di sistemi WSD Word Sense Disambiguation, il processo con il quale si precisa il significato di una parola o di un insieme di frasi, che denota significati diversi a seconda dei contesti, per evitare che sia ambigua.

MULTIWORDNET

WordNet è un sistema di gestione di un dizionario lessicale della lingua inglese basato sulle attuali teorie psicolinguistiche della memoria lessicale umana. È un database semantico-lessicale per la lingua inglese elaborato dal linguista George Armitage Miller presso l'Università di Princeton, che si propone di organizzare, definire e descrivere i concetti espressi dai vocaboli

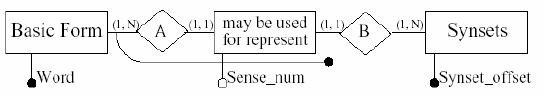
Le categorie sintattiche (nomi, verbi, aggettivi e avverbi) sono organizzate in insiemi di sinonimi che rappresentano un inerente concetto lessicale. Gli insiemi di sinonimi sono poi collegati tra loro da diversi tipi di relazioni. A ben vedere, WordNet non è semplicemente un dizionario on-line, ma non può neanche essere considerato un’ontologia, ciononostante, il numero di applicazioni che usano WordNet più come un’ontologia che come una risorsa lessicale è in crescita.

In WordNet, invece, i termini sono organizzati in concetti e non in ordine alfabetico, in base ai risultati delle ricerche psicolinguistiche.

Il punto focale della struttura di WordNet è l’associazione tra lemma (la forma scritta o il suono) e significato (il concetto associato), rappresentata in figura con una notazione E/R.

Esiste infatti un’associazione convenzionale tra la forma delle parole e il concetto che esse esprimono; tale associazione risulta di tipo molti-a-molti, e quindi dà luogo alle proprietà di:

* *Sinonimia*: proprietà di un significato di avere due o più lemmi in grado di esprimerlo. Un gruppo di lemmi tra loro sinonimi è chiamato synset.
* *Polisemia*: proprietà di un lemma di poter esprimere uno o più significati.



MultiWordNet è un database lessicale multilingue in cui il WordNet italiano è strettamente allineato con Princeton WordNet 1.6.

I synset italiani vengono creati in corrispondenza dei synset di Princeton WordNet, ove possibile, e le relazioni semantiche vengono importate dai corrispondenti synset inglesi.

MultiWordNet implementa il modello expand-model.

L’expand-model si basa sul presupposto che tra gli stessi concetti, in linguaggi diversi, intercorrono le stesse relazioni; parlando di ontologie, se due synset in wordnet sono legati da una relazione, i due synset equivalenti in un altro wordnet saranno legati dalla stessa relazione.

Questo modello appare visibilmente più semplice da implementare e garantisce maggiore compatibilità tra le strutture semantiche delle lingue rappresentate.

Infatti viene presa come base quella di Wordnet, dal quale vengono importate tutte le relazioni semantiche.

I synset una volta inseriti, vengono collegati collegati, tramite delle relazioni di equivalenza, ai synset di Wordnet.

Da un punto di vista architetturale, [MultiWordNet](https://multiwordnet.fbk.eu/english/home.php) rappresenta un’estensione della matrice lessicale di WordNet in una matrice lessicale multilingua.

Se prendiamo come base di riferimento WordNet, la costruzione della matrice lessicale multilingua con l’aggiunta dell’italiano consiste nel ri-mappare i lemmi italiani secondo i synset della lingua inglese. Il risultato è una completa ridefinizione delle relazioni lessicali (che collegano un lemma a un synset) della lingua italiana; per quanto riguarda invece le relazioni semantiche, vengono mantenute il più possibile quelle definite per l’inglese.

Analizzando la [sezione indicata,](https://multiwordnet.fbk.eu/english/mwndump.php) sono presenti informazioni che riguardano il database di MultiWorrdNet, in cui si evince sia composto da 11 file e 10 tabelle che rappresentano un dump del database MySQL e 1 file di testo.

Di seguito le info del db:

* **common\_relations.sql** lists all the semantic relations that are common to all languages

INSERT INTO common\_relations VALUES ('\*','v#00001740','v#00003763',NULL);

* **english\_relations.sql** and **italian\_relations.sql** contain the relations that are language dependent. These relations are instances of the standard lexical relations used in Princeton WordNet (e.g. antonymy, pertains to, etc.).

INSERT INTO english\_relations VALUES ('!','v#00009549','v#00009666','rest','be\_active',NULL);

* **english\_synsets.sql** and **italian\_synsest.sql** contain the English and Italian synsets (most of them are aligned with the Princeton WordNet but some are new ones)

INSERT INTO english\_synsets VALUES ('n#00008864',' plant flora plant\_life ','a living organism lacking the power of locomotion');

* **english\_index.sql** and **italian\_index.sql** contain the lists of the English and Italian lemmas. The purpose of these tables is to retrive very quickly the synset ids and the possible searches starting from a lemma in all its PoS.

INSERT INTO italian\_index VALUES ('parlamentare','#m @','@ ~',NULL,NULL, 'n#07457674',' v#00518082','a#02590962',NULL);

* **semfield.sql** contains one or more domain labels (version 1.1.1) for each synset in MultiWordNet. Domain labels are language-independent and apply to both English and Italian synsets

INSERT INTO semfield VALUES ('n#00028871','Aeronautic Tourism','Aeronautica Turismo');

* **semfield\_hierarchy.sql** contains the hierarchy of the 164 domain labels used to tag MultiWordNet synsets

INSERT INTO semfield\_hierarchy VALUES (25,'Sport','Sport','Sport','Free\_Time','Badminton Baseball Basketball Cricket Football Golf Rugby Soccer Table\_Tennis Tennis Volleyball Cycling Skating Skiing Hockey Mountaineering Rowing Swimming Sub Diving Racing Athletics Wrestling Boxing Fencing Archery Fishing Hunting Bowling');

* **english\_frame.sql** contains the subcategorization frames of the English verbs. Up to now, Italian verbs do not have this kind of information.

INSERT INTO english\_frame VALUES ('v#00182674',8,'get\_over');

* **verb-frames.txt** contains the list of the 35 different kinds of subcategorization frames for English verbs, each identified by a progressive number from 1 to 35.

1 Something s

35 Something s INFINITIVE

Una seconda fonte che riconduce al db di MultiWordNet è riconducibile a questo [link](https://github.com/frapontillo/multiwordnet-simple).

SENTIWORDNET

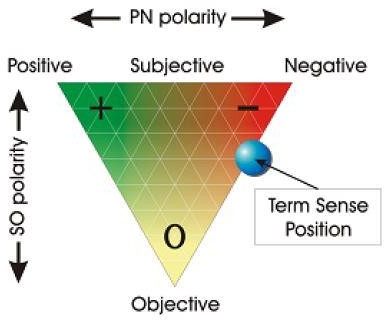
SentiWordNet (Esuli, Sebastiani, 2006) è una risorsa lessicale sviluppata da Andrea Esuli e Fabrizio Sebastiani con lo scopo di realizzare, partendo da WordNet 3.0, un lessico adatto alla sentiment analysis lexicon-based. Attualmente arrivato alla versione 3.0, SentiWordNet riporta, per ogni synset, tre punteggi di polarità, “positive", “negative”, “objective", partendo dai seguenti presupposti:

* tutti i termini appartenenti allo stesso synset hanno medesima polarità
* una stessa parola, se appartiene a synset diversi, ha diversa polarità in base al suo significato

La costruzione di SentiWordNet parte dalla classificazione dei synset, i cui punteggi sono ottenuti per ensamble learning (Russel, Norving,2005) su vari classificatori ternari "allenati" su differenti training set.

I punteggi dei synset possono essere rappresentati graficamente in un triangolo come in figura.

Gli angoli rappresentano le tre etichette (in alto a sinistra la classe positiva, in alto a destra la classe negativa, in basso la classe oggettiva), il cerchietto blu indica la posizione del significato del termine, gli assi PN-polarity e SO-polarity, infine, consentono l'interpretazione corretta della polarità e dell'oggettività del termine.



È stato possibile reperire un dataset di SentiWordNet a questo [link](https://www.kaggle.com/datasets/nltkdata/sentiwordnet?resource=download) e altre informazioni anche a questo [link](https://github.com/aesuli/SentiWordNet).

BABELNET

[BabelNet](https://babelnet.org/) è un dizionario enciclopedico multilingue innovativo, con un'ampia copertura lessicografica ed enciclopedica dei termini, e una rete/ontologia semantica che collega concetti ed entità denominate in una rete molto ampia di relazioni semantiche.

BabelNet segue il modello WordNet basato sulla nozione di synset (per insieme di sinonimi), ma lo estende per contenere lessicalizzazioni multilingue. Ogni synset BabelNetrappresenta un dato significato e contiene tutti i sinonimi che esprimono quel significato in una gamma di lingue diverse.

È presente una guida delle informazioni relative alle query di BabelNet a questo [link.](https://babelnet.org/guide#HowcanIdownloadtheBabelNetindices) Per poter scaricare il dataset è possibile effettuare l’iscrizione al [link](http://nlp.uniroma1.it/resources/login.html) indicato.

La maggior parte dei file ottenuti è possibile poterli leggere in modo chiaro con Libre Office.

NEO4J

Neo4j è un software per database a grafo open source sviluppato interamente in Java. È un database totalmente transazionale, che viene integrato nelle applicazioni permettendone il funzionamento stand alone e memorizza tutti i dati in una cartella.

È un database a grafo nativo, creato per sfruttare non solo i dati ma anche le relazioni che li caratterizzano.

La struttura fondante per i database a grafo è la "node-relationship"; la memorizzazione e la navigazione avviene utilizzando dei nodi, le loro relazioni e le loro proprietà.

I grafi rappresentano uno dei modi più efficienti e naturali di lavorare con i dati: sono estremamente intuitivi e mostrano l’interconnessione di concetti e idee.

Attraverso Neo4j è possibile superare le costruzioni ordinate dei DB relazionali, semplificando la navigazione tra i dati in memoria, promuovendo analisi in real-time e semplificando lo sviluppo di modelli rappresentativi di infrastrutture del mondo reale, servizi business, relazioni sociali, che per loro natura sono fluidi e monodimensionali.

Sia sui nodi che sugli archi è possibile porre delle "etichette" che ne descrivono il tipo; dentro ai nodi e agli archi è possibile inserire dei record con un insieme di campi e i loro relativi valori, per cui è necessario poter individuare i tipi di nodi che sono le entità e gli archi che sono le relazioni binarie, per poter stabilire le proprietà per ciascun tipo di nodo e arco.

Il database può essere usato sia in modalità embedded che server.

Nella modalità embedded si incorpora il database nell'applicazione (con Maven o includendo i file JAR) e questo viene eseguito all'interno della JVM, quindi nello stesso processo ma accettando vari thread concorrenti. Nella modalità server invece il database è un processo a sé stante a cui si accede tramite REST facendo delle query e ricevendo i dati in remoto.

La struttura a grafo di Neo4j si mostra estremamente comoda ed efficiente nel trattare strutture come gli alberi estratte ad esempio da file XML, filesystem e reti, che vengono rappresentate con naturalezza da un grafo poiché sono esse stesse dei grafi. L'esplorazione di queste strutture risulta in genere più veloce rispetto a un database a tabelle perché la ricerca di nodi in relazione con un certo nodo è un'operazione primitiva e non richiede più passaggi, in genere tre impliciti in una join di SQL, su tabelle diverse.

Neo4J supporta un linguaggio dichiarativo chiamato Cypher, progettato specificamente per interrogare i grafi ed i loro componenti.

I comandi Cypher sono basati sulla sintassi SQL e sono pensati per le query ad hoc di dati del grafo. In generale i database a grafo sono navigabili utilizzando le relazioni.

In Neo4J, nodi e relazioni supportano le “proprietà” e una coppia chiave-valore dove vengono memorizzati i dati. Questo tipo di memorizzazione e di navigazione non è possibile con un RDBMS (database relazionali) a causa delle rigide strutture di tabella e all'incapacità di seguire i collegamenti tra i dati in qualunque direzione possibile.

Cypher prende in prestito la sua struttura da SQL: le query vengono create utilizzando varie clausole. Le clausole sono concatenate e alimentano tra loro set di risultati intermedi. Ad esempio, le variabili corrispondenti di una MATCH clausola saranno il contesto in cui esiste la clausola successiva.

Il linguaggio di query è composto da diverse clausole distinte.

Di seguito sono riportati alcuni esempi di clausole utilizzate per leggere un grafo:

* MATCH: lo schema a grafo da abbinare. Questo è il modo più comune per ottenere dati dal grafo.
* WHERE: è parte di MATCH, OPTIONAL MATCH e WITH. Aggiunge vincoli a un pattern o filtra il risultato intermedio che passa attraverso WITH.
* RETURN: Cosa restituire.

La maggior parte delle volte le query Cypher stanno leggendo o aggiornando le query, che vengono eseguite su un grafo. Esistono anche comandi amministrativi che si applicano a un database o all'intero DBMS.

I comandi amministrativi non possono essere eseguiti in una sessione connessa a un normale database utente, ma devono invece essere eseguiti all'interno di una sessione connessa al database di sistema. Tutti i server Neo4j contengono un database integrato chiamato System, che si comporta in modo diverso rispetto a tutti gli altri database. Il database system memorizza i dati di sistema e non è possibile eseguire query sui grafici su di esso.

Una nuova installazione di Neo4j include due database:

* system- il database di sistema sopra descritto, contenente i metadati sul DBMS e la configurazione della sicurezza.
* neo4j- il database predefinito, denominato utilizzando l'opzione di configurazione dbms.default\_database=neo4j.

Il flusso di dati all'interno di una query Cypher è una sequenza non ordinata di mappe con coppie chiave-valore, un insieme di possibili associazioni tra le variabili nella query e i valori derivati dal database.

La versione di Neo4j che si sta utilizzando è 4.4.4.

Questo set viene perfezionato e ampliato dalle parti successive della query.

* MATCH: specificare i modelli da cercare nel database.
* OPTIONAL MATCH: specificare i modelli da cercare nel database durante l'utilizzo nulls per le parti mancanti del modello.

Per poter aggiungere il driver ad un progetto Java, si può usare Gradle o Maven, ed è sufficiente includere la dipendenza a neo4j-java-driver.

Nel caso di Maven si aggiunge nel file pom.xml.

Una volta che Maven ha aggiornato le dipendenze, possiamo connetterci al database per eseguire delle query in Cypher.

L’oggetto Driver è thread-safe. Esso è stato pensato per essere inizializzato una volta sola nel ciclo di vita dell’applicazione (nel caso di applicazione web potrebbe essere un singleton iniettato dal container), in modo che esista una sola istanza di Driver per ogni database da utilizzare.

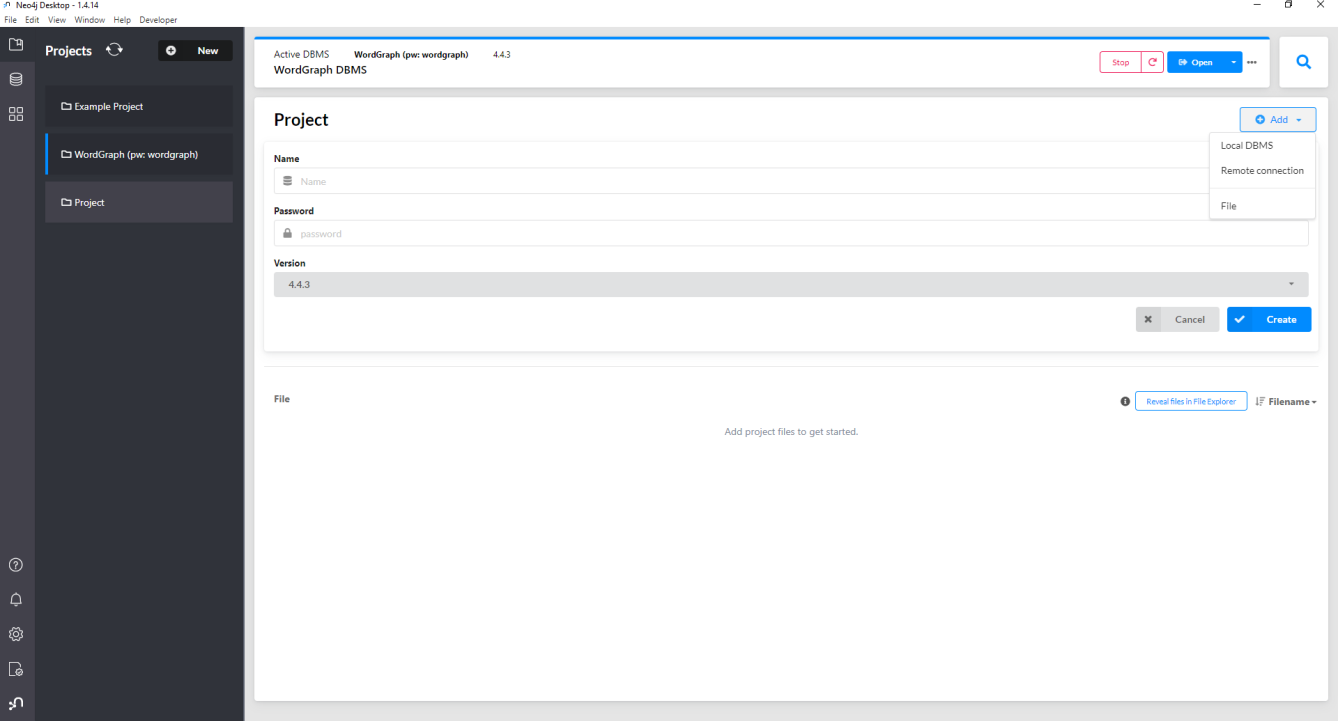
Di seguito sono indicati i passi per poter integrare Neo4j a Java.

Si dà per scontato che si installi Eclipse e [Neo4j](https://neo4j.com/download/) nelle sezioni dedicate.

Inserite le informazioni istituzionali comparirà una schermata che permetterà di far partire l’eseguibile e comparirà la Activation Key che verrà inserita nella schermata e procederà alla fase d’installazione.

La figura di seguito mostra la schermata principale di Neo4j.

1. Cliccando sul pulsante *start* si ha la possibilità di procedere all’avvio del db che è possibile creare dalla sezione laterale cliccando sul pulsante *add* > *Local DBMS* e inserire le informazioni nel nuovo progetto che conterranno il db.



1

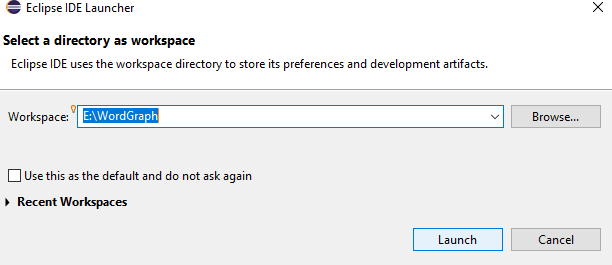
2

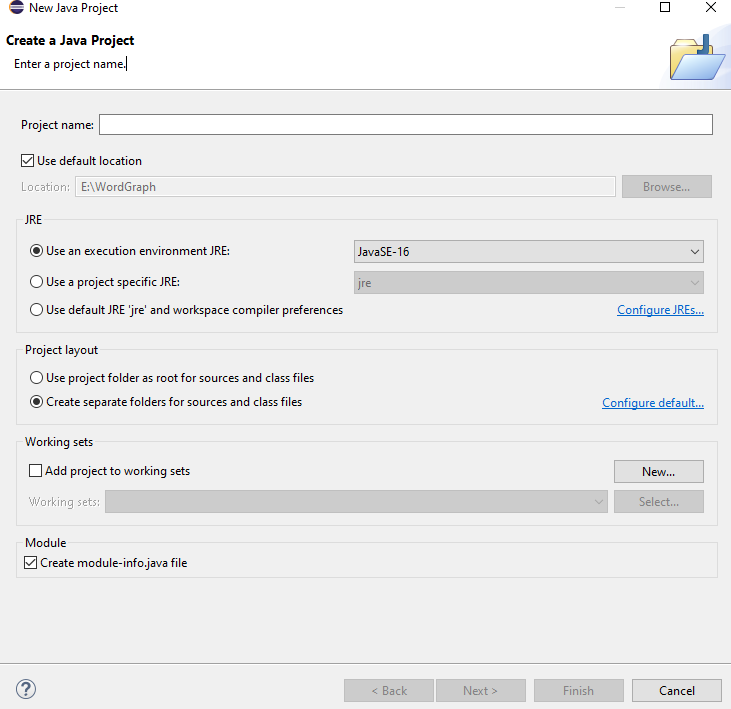
3

4

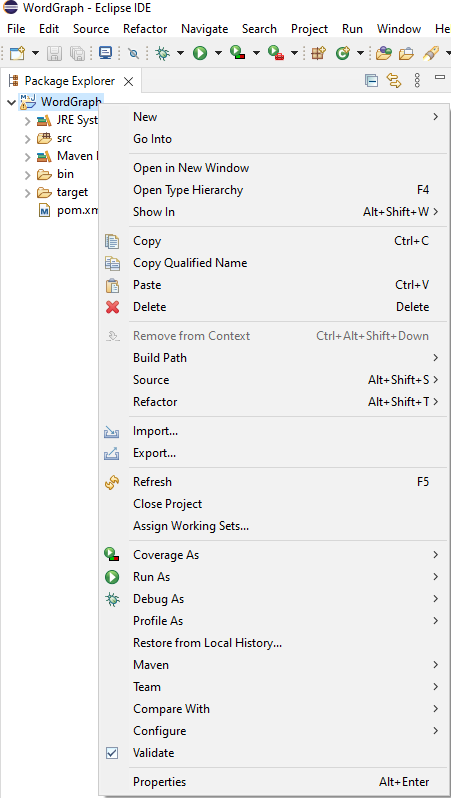
5

1. In Eclipse scegliere la directory e procedere all’inserimento del db Neo4j creando un nuovo progetto.

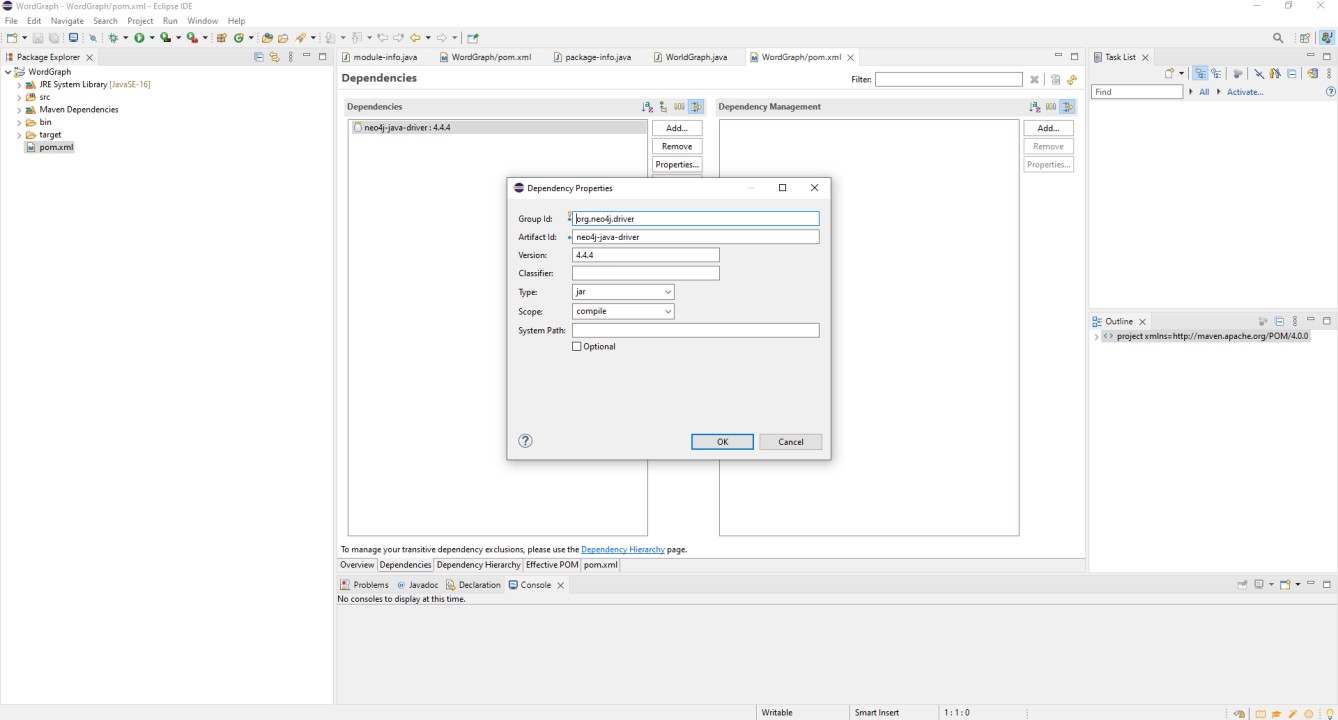




Per poter utilizzare Neo4j deve essere creato il file *pom.xml* ottenuto dalla directory principale selezionando *configure* > *convert to Maven Project*.



Successivamente aprire il file *pom.xml* e inserire le informazioni come in figura nella sezione *dependence* che è possibile visionare al [link](https://neo4j.com/developer/java/) indicato.

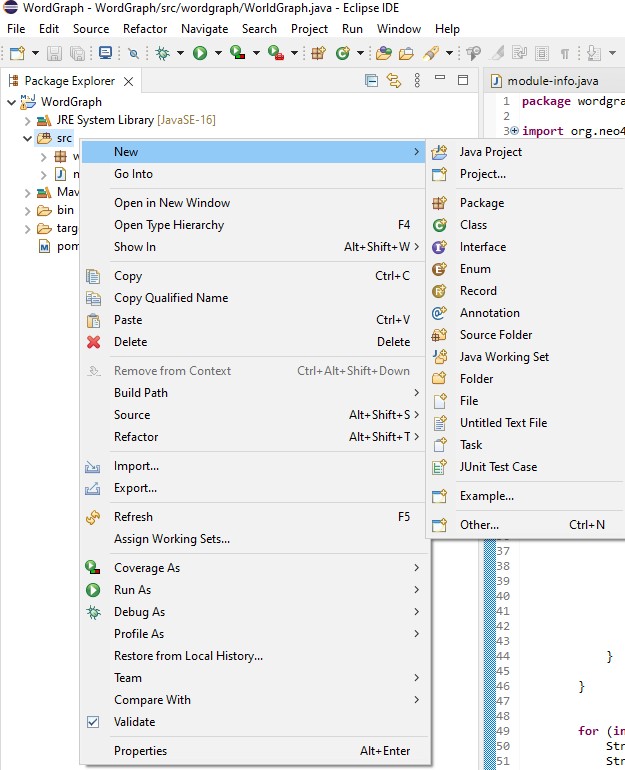


1

3

2

Nella cartella *src* con il tasto destro cliccare su *new* > *package* e successivamente inserire la classe.



ANALISI FILE

I file ottenuti dall’analisi di Wordnet, SentiWordNet, MultiWordNet sono i seguenti:

* etymwn

- etymwn-20130208

* langcode per le lingue fornitoci
* uwn\_tsv

Lo strumento utilizzato è [Libre Office.](https://it.libreoffice.org/download/download/)

Per il file **etymwn.dat** (l’estensione dat identificano generalmente file di dati generici che possono contenere qualunque tipo di informazione: immagini, fogli di testo, fogli di calcolo o file zip, o file multimediali come immagini o musica) non è stato possibile analizzarlo perché in formato incomprensibile e non letto da nessun software.

Per il file **etymwn-20130208** sono presenti 2 file

* readme.txt
* **etymwn.tsv**

Dalla codifica visionata con Libre Office del file etymwn.tsv emerge quanto segue:

possiedo una parola, possiedo la lingua della parola, e ho una relazione con un’altra parola della stessa lingua e la relazione di tipo diverso.

* rel: etymological\_origin\_of
* rel: has\_derived\_form
* rel:is\_derived\_from
* rel: etymology
* rel: variant:orthography
* rel: etymologically\_related

Dalla codifica del file **langcodes.csv** con Libre Office è possibile visionare l’elenco delle lingue. Codici delle lingue, con annessi nomi delle lingue tradotti.

Dalla codifica del file **uwn\_tsv** sono presenti in file .zip due file e l’analisi viene fatta sul file

**Uwn-dump\_201012** con Libre Office. Universal WordNet

Relazioni:

* rel:lexicalization
* rel:means

BOZZA DB

Nodi:

* lemma
* synset

Relazioni

* lemma - "is in" - > synset
* synset - "relation" - > synset
* synset - "contains" - > lemma

Attributi:

* per i synset: gloss, tipo (v, n), id
* per i lemma: language

La relation tra synset è definita in questa lista:

Verb relations

Semantic: ! antonym @ hypernym ~ hyponym \* entailment > causes ^ also-see $ verb-group | nearest Lexical: ! antonym @ hypernym ~ hyponym +c composed-of -c composes derived-from / related-to

Adjective relations

Semantic: ! antonym & similar-to = is-value-of ^ also-see | nearest

Lexical: ! antonym < participle +c composed-of -c composes derived-from / related-to.

BOZZA DELLA STRUTTURA DEL DATABASE

Link: <https://pypi.org/project/multiwordnet/>

Nodi:

* lemma
* synset

Relazioni

* lemma - "is in" - > synset
* synset - "relation" - > synset
* synset - "contains" - > lemma

Attributi:

* per i synset: gloss, tipo (v, n), id
* per i lemma: language

La relation tra synset sono definite in questa lista:

Verb relations

Semantic: ! antonym @ hypernym ~ hyponym \* entailment > causes ^ also-see $ verb-group | nearest

Lexical: ! antonym @ hypernym ~ hyponym +c composed-of -c composes derived-from / related-to

Adjective relations

Semantic: ! antonym & similar-to = is-value-of ^ also-see | nearest

Lexical: ! antonym < participle +c composed-of -c composes derived-from / related-to.

MULTIWORDNET

Nodi

* + :Synset

Rappresenta il synset all’interno di WordNet

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome attributo** | **Descrizione**  **Attributo** | **Tipo attributo** | **“Mandatory”**  **T/F** |
| id | Synset id | String | TRUE |
| word | Parole del  synset separate da spazi | String | TRUE |
| phrase |  | String | FALSE |
| gloss | Spiegazione del  synset | String | FALSE |
| language | Lingua del  synset | String | TRUE |
| Positive sentiment |  | Float | FALSE |
| Negative sentiment |  |  | FALSE |
| Neutral sentiment |  |  | FALSE |

* + :Lemma

Rappresenta le parole all’interno di WordNet

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome attributo** | **Descrizione**  **Attributo** | **Tipo attributo** | **“Mandatory”**  **T/F** |
| id | Synset id | String | TRUE |
| pos | Tipo parola 'noun','verb','adj ective','adjective satelite','adverb'  (‘n’,’v’,’a’,’s’,’r ’) | String | TRUE |

|  |  |  |  |
| --- | --- | --- | --- |
| lemma | La parola | String | TRUE |
| is\_phrase | Se lemma e’ una  frase | Y/N | TRUE |
| language | Lingua del  synset | String | TRUE |

* + :SemanticField

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome attributo** | **Descrizione**  **Attributo** | **Tipo attributo** | **“Mandatory”**  **T/F** |
| id | Synset id | String | TRUE |
| english | Semantic field  in inglese | String | TRUE |

* + Archi

|  |  |
| --- | --- |
| **Relazione (tra synset)** | **Simbolo** |
| Antonym | ! |
| Hypernym | @ |
| Hyponym | ~ |
| Entailment | \* |
| Causes | > |
| Also see | ^ |
| Verb-Group | $ |
| Nearest | | |
| Composed of | +c |
| Composes/Derived from | -c |
| Derived from |  |
| Related to | / |
| Member of | #m |
| Substance of | #s |
| Part of | #p |
| Has member | %m |
| Has substance | %s |

|  |  |
| --- | --- |
| Has part | %p |
| Has role | +r |
| Is role | -r |
| Similar to | & |
| Is value of | “=” |
| Participle | < |
| **Relazione (tra synset e semfield)** |  |
| Belongs (?) | -------------------------------- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Il proseguire del progetto si articola come segue:

* + si è effettuato l’upload del database a grafo con le informazioni di Wordnet;
  + si è effettuato l’upload della lingua inglese con successiva implementazione della lingua italiana;
  + è stato creato un db con delle informazioni indicate come segue:

*synset - informazione sul synset, contenenti id e gloss*

*(nodi fatti in modo tale che per synset comuni alle varie lingue, risultino stessi id)*

*lemma - nodo che rappresenta la singola parola appartenente ad un synset. (nodi sono collegati a nodi synset tramite la relazione word2synset*)

* + sono state implementati i file langcodes che rappresenta una lista di codici linguistici;
  + per poter implementare le informazioni di SentiWordnet, è stato seguito un mapping tra il vecchio Wordnet 1.6 e il nuovo Wordnet 3.0, in cui tra alcuni synset sono specificati degli score inerenti alla Sentiment analysis;
  + il db che si possiede contiene quindi le informazioni di diverse lingue allineate a Wordnet 3.0;
  + il db di Wordnet evita di caricare duplicati dei nodi;
  + sono state uplodate le lingue di MultiWordNet come segue: inglese, italiano, francese, spagnolo, portoghese, ebreo e latino;
  + sono state uplodate le lingue di UWN che comprende e sono state create le relazioni con i synset di MultiWordNet;
  + sono state uplodate i sentiment scores dei vari termini di SentiWordnet.