Esonero/Homework 3 - Metodologie 2019
Corso di Laurea Triennale in Informatica - Sapienza
prof. Roberto Navigli – A.A. 2018/2019
(Esonero 2 per gli studenti in teledidattica)

Importante:
per favore commentate selezionando il testo di interesse -> tasto destro -> commento.

WordNet è un lessico elettronico (una sorta di dizionario evoluto) della lingua inglese disponibile all'URL https://wordnet.princeton.edu/. Potete provarlo all'indirizzo: http://wordnetweb.princeton.edu/perl/webwn (si noti tuttavia che questa versione, la 3.1, non è scaricabile direttamente). Al contrario dei dizionari tradizionali, che organizzano la conoscenza lessicografica in ordine alfabetico, WordNet è incentrato sul concetto di synset (*synonym set*), ovvero l'insieme dei sinonimi che sono utilizzati per esprimere il concetto. Ad esempio, { subject, topic, theme } sono i sinonimi utilizzati per esprimere il concetto di "argomento", mentre { theme, melodic theme, musical theme, idea } sono i sinonimi che esprimono il concetto di tema musicale. Si noti che i due synset condividono una parola, *theme*, a indicare che essa può assumere più di un significato (è, cioè, polisemica). WordNet fornisce le seguenti informazioni per un synset:

- nun offset (o id) del synset
- la parte del discorso (nome, verbo, aggettivo o avverbio) che, insieme all'offset, può essere utilizzato come id univoco del synset (es. `00001740n`)
- i sinonimi che esprimono tale concetto (es. `subject, topic, theme`)
- una (e una sola) definizione testuale detta glossa (es. `the subject matter of a conversation or discussion`)
- (opzionalmente) uno o più esempi d'uso dei sinonimi del synset (es. *`"he didn't want to discuss that subject"; "it was a very sensitive topic"; "his letters were always on the theme of love"`*)
- relazioni verso altri synset (relazioni semantiche, ad esempio { car, auto, automobile, machine, motorcar } ha come iperonimo, ovvero generalizzazione, { motor vehicle, automotive vehicle }). Qui trovate un elenco dei tipi di relazioni disponibili in WordNet: https://wordnet.princeton.edu/documentation/wninput5wn. Le relazioni possono essere codificate mediante stringhe (ovvero utilizzando gli stessi simboli usati nei file data di WordNet, es. @ per l'iperonimia) o usando un'enumerazione che implementa un'interfaccia `WordNetRelation` (**è richiesta l'implementazione di entrambe le opzioni**). Grazie a una qualsiasi delle due soluzioni è possibile ammettere anche altre relazioni in futuro.

I synset codificano le 4 parti "aperte" del discorso (ovvero, quelle in cui è possibile creare nuove parole), ovvero: nomi, verbi, aggettivi e avverbi.

Negli anni, sono state rese disponibili diverse versioni, aggiungendo nuovi synset, ma anche modificando quelli esistenti (ad esempio, eliminando, fondendo o separando alcuni synset, modificandone le definizioni, ecc.). I pacchetti con le diverse versioni di WordNet sono

disponibili qui (non è necessario gestire la cartella `extra`). Si noti che possiamo vedere una versione di WordNet come un grafo i cui nodi sono i synset e i cui archi sono le relazioni semantiche tra i nodi.

L'obiettivo dell'homework 3 è sviluppare un insieme di classi finalizzate a:

- **18 punti:** caricare un'intera versione di WordNet in memoria, iterabile sui synset, mediante il metodo `WordNet.getInstance`. Alla chiamata di `getInstance` con lo stesso input (la stringa della versione, ad esempio "1.6" o "3.0") il metodo restituisce il medesimo oggetto contenente il dizionario caricato la prima volta per la stessa versione richiesta. Il file contenente i synset e tutte le loro informazioni si chiama `data.<POS>` (con `<POS> = noun, verb, adj e adv` secondo la parte del discorso). Il formato del file `dict/data.<POS>` è illustrato qui (notate che alcuni campi sono in esadecimale!): https://wordnet.princeton.edu/documentation/wndb5wn Potete tralasciare: `lex_id, source/target e frames`. Si noti che `ss_type` nel file `data.<POS>` può assumere anche adjective_satellite, che va considerato aggettivo a tutti gli effetti.
- **16 punti:** fornire una mappatura tra versioni diverse (ovvero, una successiva all'altra) di WordNet mediante la classe `Mapper`. La mappatura può essere effettuata con qualsiasi algoritmo, ma come minimo **(fino a 12 punti dei 16)** deve mappare correttamente i synset identici (stessa glossa, stessi sinonimi). Ad esempio, nel caso di una mappatura da WordNet 1.6 a WordNet 3.1, entrambe le versioni hanno i 5 seguenti significati in comune per il sostantivo *subject* e forniscono definizioni identiche:

  (n) **subject**, topic, theme (the subject matter of a conversation or discussion) *"he didn't want to discuss that subject"; "it was a very sensitive topic"; "his letters were always on the theme of love"*
  (n) **subject**, content, depicted object (something (a person or object or scene) selected by an artist or photographer for graphic representation) *"a moving picture of a train is more dramatic than a still picture of the same subject"*
  (n) topic, **subject**, issue, matter (some situation or event that is thought about) *"he kept drifting off the topic"; "he had been thinking about the subject for several years"; "it is a matter for the police"*
  (n) **subject**, case, guinea pig (a person who is subjected to experimental or other observational procedures; someone who is an object of investigation) *"the subjects for this investigation were selected randomly"; "the cases that we studied were drawn from two different communities"*
  (n) national, **subject** (a person who owes allegiance to that nation) *"a monarch has a duty to his subjects"*

  A due synset uguali in versioni diverse corrisponde una mappatura con punteggio pari a 1.0. Tuttavia, le due versioni differiscono nei sinonimi nel senso di materia:
  v 1.6: (n) discipline, **subject**, subject area, subject field, field, field of study, study, *branch_of_knowledge* (a branch of knowledge)

v 3.1: (n) discipline, **subject**, subject area, subject field, field, field of study, study, *bailiwick* (a branch of knowledge)

e nella definizione del synset { subject } nel senso di soggetto grammaticale:

v1.6: (n) **subject** ((linguistics) the grammatical constituent about which something is predicated in a sentence)

v3.1: (n) **subject** ((grammar) one of the two main constituents of a sentence; the grammatical constituent about which something is predicated)

tuttavia le definizioni condividono alcune parole, come *sentence*, *predicated* e *something*. Il mappatore può utilizzare queste parole per determinare la similarità tra i due synset (con un valore strettamente < 1.0 ma > 0.0 la cui formula lo studente è libero di sviluppare) che non hanno match perfetto e mapparli anche se hanno definizioni differenti. Infine, la versione 3.1 fornisce un ottavo synset che non ha un corrispondente nella versione 1.6 e che quindi non sarà mappato:

(n) **subject** ((logic) the first term of a proposition).

La classe `WordNet` deve disporre dei seguenti metodi:

- `getInstance`, che - data in input una versione - restituisce la corrispondente istanza di WordNet. Se la versione non è disponibile nella cartella wordnet-releases, il metodo restituisce `null`.
- `getSynsets`, in due versioni: una prende in input un lemma (ovvero, una parola nella sua forma base, se composta da più parole si assume che le parole siano separate da _) e una seconda versione che prende in input sia il lemma che la parte del discorso. Entrambe restituiscono la lista di synset che contengono il lemma in questione (la seconda filtrando sulla parte del discorso). Entrambe le versioni devono garantire di restituire ALMENO i synset i cui lemmi fanno match esatto (es. se cerco Obama devo trovare il synset contenente “Obama”; se cerco “obama” non è strettamente necessario trovare il synset contenente “Obama” (tuttavia è consigliabile))
- `getSynsetFromID` che, dato in input l’ID sotto forma di stringa (offset+pos, es. `00001740n`), restituisce il synset corrispondente, `null` se non presente.
- `stream`, che restituisce uno stream di `Synset`
- `getVersion`, che restituisce la versione della WordNet
- `getRelatedSynsets` che, dato in input un synset e una relazione sotto forma di `WordNetRelation` o `String`, restituisce una collezione di synset correlati (ad esempio, `getRelatedSynsets(sourceSynset, “@”)` restituisce gli iperonimi del synset (es. `{ motor vehicle, automotive vehicle }` se il primo synset di car viene dato in input). Se la relazione non esiste o non ci sono istanze di tale relazione, viene restituita una collezione vuota.

La classe `Synset` deve disporre dei seguenti metodi:

- `getSynonyms`, che restituisce l'insieme dei sinonimi del synset
- `contains` che, data in input una parola, restituisce `true` se il synset contiene tra i suoi sinonimi quella parola, `false` altrimenti
- `getGloss`, che restituisce la glossa del synset
- `getExamples`, che restituisce l'insieme degli esempi del synset (vuoto se non ci sono esempi)
- `getOffset()`, che restituisce l'offset ("01234567") del Synset
- `getPOS()`, che restituisce l'oggetto enum del POS corrispondente
- `getID()`, che restituisce l'ID (offset + pos) del Synset; per pos si intende uno dei caratteri 'n', 'v', 'a', 'r' corrispondente al POS del synset.

La enum **`POS`** deve contenere le 4 parti del discorso chiamate **`NOUN`**, **`VERB`**, **`ADJECTIVE`**, **`ADVERB`** `(N.B.: ADJECTIVE include anche il tipo 's', oltre che 'a').`

La classe **`WordNetMapping`** crea la mappatura tra due versioni di WordNet (una sorgente e una di destinazione) e deve avere almeno i seguenti metodi:

- `getMapping(Synset src)`, che dato un Synset nella versione sorgente, restituisce un `Optional` contenente il `SynsetPairing` che mappa il synset sorgente con il miglior synset nella versione di destinazione

La classe **`SynsetPairing`** rappresenta la coppia `Synset` sorgente e corrispondente `Synset` destinazione con associato lo score di confidenza della coppia e deve avere almeno i seguenti metodi:

- `getSource()`, che restituisce il Synset della versione WordNet sorgente
- `getTarget()`, che restituisce il Synset della versione WordNet di destinazione
- `getScore()`, che restituisce un valore (double) di confidenza della coppia compreso tra 0.0 e 1.0

La classe `Mapper` è dotata di un metodo `map` che, presi in input due `WordNet` (la prima sorgente, la seconda destinazione) restituisce un oggetto di tipo `WordNetMapping` che fornisce le mappature dalla wordnet di partenza a quella di destinazione (si noti che l'inverso potrebbe avere mappatura leggermente differenti).

Il codice dovrà essere progettato in modo tale che sia possibile eseguire il main della seguente classe:

```
package it.uniroma1.metodologie2019.hw3;

// aggiunge import

public class MapperTest
{
	public static void main(String[] args)
	{
		WordNet wn21 = WordNet.getInstance("2.1");
```

```
            WordNet wn30 = WordNet.getInstance("3.0");
            List<Synset> synsets1 = wn30.getSynsets("house", POS.VERB);
            System.out.println(synsets1);
            List<Synset> synsets2 = wn30.getSynsets("run");
            System.out.println(synsets2);

            wn21.stream()
                  .filter(s -> s.getPOS() == POS.NOUN)
                  .forEach(s ->
            System.out.println(s.getSynonyms()+"["+s.getOffset()+"]:"+s.ge
            tGloss()));
            WordNetMapping map = Mapper.map(wn21, wn30);
            wn21.stream()
                  .map(map::getMapping)
                  .flatMap(Optional::stream)
                  .forEach(m ->
            System.out.println(m.getSource()+"->"+m.getTarget()+":"+m.getS
            core()));
      }
}
```

Inoltre, il codice deve superare i test presenti nel seguente file di JUnit: https://drive.google.com/file/d/10eJcM4cqn5ovdgSwDnxQoN8cJjjU8r0M/view?usp=sharing
La classe va posizionata all'interno del package it.uniroma1.metodologie2019.hw3 e deve essere possibile eseguirla senza fare ulteriori import.

Checklist prima della consegna

- Il progetto si trova all'interno del package `it.uniroma1.metodologie2019.hw3` (è possibile utilizzare sottopackage, ma le classi fondamentali utilizzate nel main sopra devono trovarsi in quel package)
- Le costanti (come ad esempio i percorsi o i nomi di file) devono essere specificate come costanti pubbliche e statiche in cima alla classe o all'interfaccia di riferimento.
- Si deve assumere che la cartella `wordnet-releases` si trovi alla radice del progetto, ovvero allo stesso livello di `src` e `bin`. **Tuttavia, la cartella con i dati NON DEVE ESSERE CONSEGNATA NEL PROGETTO GITHUB. La aggiungeremo noi in fase di test.**
- Utilizzare `Path` per memorizzare i percorsi e `Files.lines` per leggere le righe del file oppure `Files.newBufferedReader`.
- Verificare la visibilità di ogni campo (tendenzialmente privata)
- Verificare di aver generato il javadoc del progetto (tasto destro sul progetto, export, javadoc)

Elementi di valutazione

- Corretta compilazione dell'esempio fornito sopra (richiesto per ottenere la sufficienza)
- Corretto caricamento di WordNet e mappatura dei synset identici (richiesto per ottenere la sufficienza)
- La qualità e la pulizia del codice e il suo orientamento a oggetti
- L'utilizzo delle strutture dati appropriate, anche nell'ottica dell'ottimizzazione e dell'uso della memoria
- L'utilizzo di stream
- Documentazione javadoc per commentare i package (file package-info.java nella radice del package), le classi, i metodi e i campi
- Algoritmo di mappatura

Modalità di consegna

La consegna deve essere effettuata mediante il progetto `metodologie2019_homework3` all'interno del proprio account github. E' necessario compilare il seguente modulo ai fini della consegna. La scadenza è prevista per la **mezzanotte in qualsiasi parte del pianeta del 13 maggio 2019**. **I progetti devono essere privati e condivisi esclusivamente con l'utente babelnet.**

Sfide

- Mappare correttamente synset che, tra versioni differenti, hanno differenze non ovvie, ma neanche impossibili da riconciliare **(+3 punti sui 12 punti base del mapper).**
- Risolvere automaticamente o comunque identificare gli errori presenti nella separazione tra glosse ed esempi **(+1 punto sui 12 punti base del mapper).** Ad esempio, nella versione 1.6, uno dei sensi di adversity.n è definito come: `"a stroke of ill fortune; a calamitous event: "a period marked by adversities"`. L'uso dei due punti è errato, in quanto per separare la definizione dagli esempi si utilizza il punto e virgola.

I migliori homework saranno premiati con una maglietta di BabelNet!

Plagio

Indicazioni significative di plagio porteranno gli studenti coinvolti all'annullamento di tutti gli esoneri svolti (anche in precedenza), **senza distinguere tra chi ha fatto copiare e chi ha copiato**. Si consiglia **caldamente di non condividere il codice con altri studenti (neanche per 5 minuti)**.

Piccolo compendio sulla gestione dei file

Utilizzando la classe `java.io.File` pre-Java 7:

- `File.isFile` e `.isDirectory` per sapere se si tratta di un file o di una cartella
- `File.listFiles` restituisce un array di File contenuti all'interno dell'oggetto (cartella) su cui è invocato. Ad esempio, `new File(PERCORSO_CARTELLA).listFiles()` restituisce gli oggetti `File` corrispondenti ai file contenuti all'interno della cartella.
- Per leggere un file, potete usare il seguente codice:

```
try(BufferedReader br = new BufferedReader(new FileReader(file)))
{
      while(br.ready())
      {
            // leggi ciascuna riga con br.readLine(),
            // che restituisce una stringa con la linea di testo
      }
}
catch(IOException e)
{
      // gestisci l'eccezione di I/O
}
```

Lettura dei file in Java 7-8 mediante il package `java.nio.file`:

- Potete usare la classe `java.nio.file.Files` per ottenere un oggetto di tipo `BufferedReader` che vi permette di leggere un file di testo. Per farlo, potete usare il metodo statico `Files.newBufferedReader` che richiede in input un `Path`. Il metodo vi restituisce comunque un `BufferedReader` che va gestito con il `try catch` come sopra.
- Per ottenere il `java.nio.file.Path` di un file o di una cartella (l'equivalente Java 8 di un `File`) potete usare il metodo statico della classe `Paths`: `Paths.get(PERCORSO)` (utile: c'è una versione di `Paths.get` che permette di specificare una sequenza di cartelle, che verranno concatenate col separatore corretto (`File.separator`), così come `Path` dispone di un metodo resolve per spostarsi all'interno della prossima cartella (ad es. `Paths.get("prima").resolve("seconda")` restituisce il path come se l'avessi ottenuto da `Paths.get("prima", "seconda")` o `Paths.get("prima"+File.separator+"seconda")`)

- Quando avrete visto gli stream, utilizzerete il metodo `Files.lines` che restituisce uno stream di righe del file (molto comodo, anche per l'homework!)
- Potete sempre passare da `Path a File con il metodo Path.toFile` (viceversa, da `File a Path` con `File.toPath`)
- Se dovesse servirvi di scrivere un file, è tutto come sopra, ma `Files.newBufferedWriter` o `new BufferedWriter(new FileWriter(PERCORSO))`.
- Per ottenere l'elenco dei file di una cartella, utilizzate `Files.list(Path cartella)`