Alma Mater Studiorum - Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Magistrale in Ingegneria Informatica
Fondamenti di Intelligenza Artificiale M

Risoluzione di semplici problemi aritmetici espressi in linguaggio naturale con tecniche di Intelligenza Artificiale

Tesi di laurea di:

Francesco Burelli

Relatore:

Chiar.ma Prof.ssa **Paola Mello**

Correlatori:

Ing. **Federico Chesani**

Chiar.ma Prof.ssa **Michela Milano**

Anno Accademico 2015/2016
Sessione III

Indice

Introduzione

Nell'ambito dell'Intelligenza Artificiale uno dei problemi aperti e più difficile da risolvere è la comprensione del linguaggio naturale. La complessità sintattica e la conoscenza che bisogna avere per comprendere riferimenti, relazioni e concetti impliciti rendono questo problema molto interessante e la sua risoluzione di grande importanza per lo sviluppo di applicazioni che possano interagire in modo diretto con le persone.

Questo lavoro di tesi non pretende di studiare e trovare una soluzione completa al suddetto problema, ma si prefigge come obiettivo quello di comprendere e risolvere problemi matematici di tipo logico ed aritmetico scritti in lingua inglese. La difficoltà del lavoro si riduce in quanto non si devono considerare gli infiniti ambiti conoscitivi e può concentrarsi su un'unica interpretazione del testo: quella matematica. Nonostante questa semplificazione il problema da affrontare rimane di grande difficoltà poiché è comunque necessario confrontarsi con la complessità del linguaggio naturale.

Esempi di problemi matematici che si intende risolvere si possono trovare presso il sito web dell'Università della Bocconi nella sezione dei Giochi Matematici[12]. Questi problemi richiedono la conoscenza di concetti di logica, insiemistica e di algebra lineare per essere risolti. Di seguito un esempio di quesito matematico:

> "Dieci amici organizzano una gara di tiro alla fune. I pettorali sono numerati da 1 a 10. I partecipanti si dividono in due squadre: i Potenti e gli Smilzi. Il prodotto dei pettorali dei componenti della squadra dei Potenti è un numero uguale alla somma

dei pettorali dei componenti della squadra degli Smilzi. Qual è questo numero?"

Il modello matematico che descrive questi problemi non è complesso ed una volta dedotto correttamente è di facile risoluzione tramite un risolutore automatico. La difficoltà consiste nel comprendere correttamente il testo ed estrapolarne il giusto modello.

Il lavoro che si andrà ad esporre nel seguito parte dall'analisi del testo ed arriva fino alla risoluzione del quesito matematico. Si parte quindi da un'analisi del testo con scopo generale ed una sua trasformazione in una forma più facilmente processabile che sarà soggetta ad una seconda analisi, più specifica, volta alla costruzione del modello matematico che andrà poi risolto da un risolutore automatico che ne restituirà il risultato finale.

La tesi è stata organizzata nel seguente modo.

Nel Capitolo 1, verrà data una panoramica su quali approcci esistono per l'analisi del testo, in particolare metodi simbolici e statistici, l'uso di questi nell'ambito in esame e quindi gli svantaggi e i vantaggi delle diverse metodologie. Verranno discussi alcuni progetti di ricerca esistenti volti alla risoluzione di problemi matematici descritti in linguaggio naturale, analizzando i modi con cui hanno affrontato le problematiche e i limiti delle realizzazioni.

Nel Capitolo 2, si descriverà in modo approfondito quali tipi di problemi si intende risolvere e con quali strategie. Verranno discussi gli approcci scelti per l'analisi del linguaggio naturale e la modellazione matematica dei quesiti. Infine verrà fatta una panoramica sull'architettura del sistema complessivo evidenziando e descrivendo i macro passi necessari alla risoluzione del problema.

Con il Capitolo 3 inizia la discussione sulla parte di sviluppo, dove verranno discusse le scelte progettuali del sistema e si vedrà come viene affrontato il tema dell'analisi del testo a basso livello. Non si fa ancora riferimento al dominio di interesse, si cerca invece di ottenere un modello generico del testo da elaborare nei passaggi successivi.

Nel Capitolo 4 si illustrerà in modo dettagliato gli aspetti implementativi del sistema, quali tecniche e quali pattern dell'ingegneria del soft-

ware sono stati utilizzati per rendere questa applicazione il più possibile modulare, flessibile ed estendibile. Infine saranno spiegati alcuni algoritmi e metodologie per affrontare il problema semantico e come questo viene risolto per poter essere usato efficacemente dai componenti adibiti all'interpretazione matematica.

Il Capitolo 5 descriverà il primo modulo sviluppato per la deduzione del modello matematico. Questo modulo è specifico per la tipologia di problemi logico-matematici. Il suo modello sarà di conseguenza un modello logico descritto dal linguaggio Prolog.

Nel Capitolo 6 verrà presentato l'altro modulo realizzato, dedicato alla modellazione di problemi di tipo aritmetico. La rappresentazione del modello è in Java e, oltre ai concetti matematici, fa uso del concetto di stato, indispensabile per rappresentare situazioni che mutano nel tempo.

Nel Capitolo 7 si andranno ad analizzare i risultati dei test effettuati su entrambi i moduli sviluppati, si discuteranno i punti deboli del sistema andando a studiare quando e perché il sistema può fallire. Il capitolo si concluderà con delle considerazioni su come è possibile procedere per risolvere le limitazioni del sistema, cosa migliorare del lavoro esistente e quali strategie potrebbero essere adottate in aggiunta per aumentarne le prestazioni.

In conclusione una discussione sul lavoro di tesi nel suo complesso, le problematiche risolte parzialmente o non risolte e le possibili soluzioni. Si parlerà di come è possibile estendere il sistema per migliorarne i risultati ed aumentarne le funzionalità.

Capitolo 1

Stato dell'arte

In questo primo capitolo verranno dapprima forniti i concetti base dell'analisi del linguaggio naturale, ovvero la teoria sui linguaggi e le metodologie più comuni per affrontare questo tipo di problemi. Verranno presentati alcuni strumenti per NLP (Natural Language Process) in particolare prima quelli per l'analisi sintattica e, a seguire, quelli per l'analisi semantica. Per concludere una discussione su alcuni progetti esistenti che risolvono, in modi differenti, problemi matematici descritti in linguaggio naturale.

Nella prossima sezione una breve panoramica sui concetti chiave della teoria dei linguaggi e successivamente l'applicazione al linguaggio naturale.

1.1 Teoria del linguaggio

Il linguaggio naturale, seppur molto complesso, viene tipicamente processato come si fa per un normale linguaggio formale. Prima di procedere è necessario quindi sapere che cos'è un linguaggio, da cosa è definito e come viene generato.

Un linguaggio è costituito da un alfabeto ovvero un insieme di simboli che andranno a comporre le frasi del linguaggio. Le frasi sono strutturate secondo una sintassi a sua volta definita da una grammatica. La gram-

matica è l'insieme delle regole (produzioni) che, legando i simboli non terminali a quelli terminali, danno una struttura sintattica al linguaggio.

Dato un alfabeto a,b,c, prendiamo ad esempio la seguente grammatica:

$S \rightarrow N\ V$

$N \rightarrow D\ M \mid M$

$V \rightarrow U\ N$

$D \rightarrow a$

$M \rightarrow b$

$U \rightarrow c$

Per convenzione, i simboli in maiuscolo sono simboli non terminali, ovvero quei simboli che saranno soggetti a sostituzione e non compariranno nel linguaggio e che servono solo a descrivere le regole grammaticali. Invece, i simboli in minuscolo sono i simboli terminali che costituiscono l'alfabeto e compariranno nelle frasi del linguaggio. Il simbolo S viene detto scopo e rappresenta una frase corretta. Nella grammatica in esame, una frase del linguaggio è sempre costituita dalla produzione N concatenata alla produzione V. Seguendo le regole grammaticali è facile dedurre che le possibili frasi di questo linguaggio sono, ad esempio, "abcab", "bcab", "bcb", etc. .

Esistono diversi tipi di linguaggi, ma la trattazione della teoria dei linguaggi esula dallo scopo di questa tesi, basti sapere che il linguaggio naturale viene solitamente trattato come un linguaggio di tipo "context-free", ovvero avente una grammatica libera da contesto. Ciò significa, operativamente, che la parte sinistra delle produzioni ha sempre un solo simbolo non terminale. Ciò rende la generazione, il riconoscimento e l'interpretazione del linguaggio più efficiente a livello computazionale.

Di seguito cercheremo di applicare la teoria appena vista al linguaggio naturale. Prendiamo come esempio una frase molto semplice: "Il gatto mangia il topo". Facendo l'analisi logica risulta che la frase è composta da

soggetto, verbo e complemento oggetto, a sua volta il soggetto è costituito da un articolo seguito da un nome, stessa cosa per il complemento oggetto. Possiamo quindi provare a definire una semplice grammatica che descriva la sintassi di questa frase:

$S \rightarrow SG\ V\ C$

$SG \rightarrow E$

$V \rightarrow$ mangia

$C \rightarrow E$

$E \rightarrow A\ N$

$A \rightarrow$ il

$N \rightarrow$ gatto | topo

Leggendo le produzioni si può capire come una frase sia composta da soggetto (SG), verbo (V) e complemento (C), a loro volta il soggetto e il complemento sono generati dalla produzione E che è costituita da un articolo (A) ed un nome (N). Ci si rende subito conto che questa grammatica non può descrivere tutto il linguaggio naturale e necessità di molte aggiunte. Ad esempio la produzione A, che indica l'articolo, si potrebbe arricchire in $A \rightarrow$ il | la | i | gli | le | un | una . Il complemento, quando non è complemento oggetto, solitamente è introdotto da una preposizione quindi sarebbe il caso di aggiungere la produzione per la preposizione ed un'alternativa per il complemento. Di seguito la grammatica rivista:

$S \rightarrow SG\ V\ C$

$SG \rightarrow E$

$V \rightarrow$ mangia

$C \rightarrow P\ E \mid E$

$E \rightarrow A\ N$

A → il | la | i | gli | le | un | una

N → gatto | topo

P → di | a | da | in | con | su | per | tra | fra

Purtroppo la grammatica sopra riportata ha ancora troppe semplificazioni: alcune parti della frase possono non esserci come il soggetto quando è sottinteso e il complemento oggetto quando il verbo non è transitivo. Le maggiori problematiche che emergono da questa prima, breve, analisi riguardano le parti opzionali ovvero parti di frasi che possono non esserci. Una frase assolutamente lecita può essere formata da una sola parola, ad esempio: "Mangia!".

Andiamo ora a complicare ulteriormente le cose prendendo in considerazione frasi composte da più proposizioni, coordinate e subordinate. Dall'analisi logica si evince che ad ogni predicato corrisponde una proposizione e queste possono essere legate in modo coordinato o subordinato. Prendiamo in esame le proposizioni coordinate, queste vengono legate da una congiunzione quale "e", "o" o dalla virgola. In più consideriamo anche che il soggetto ed il complemento possono essere un insieme di entità legate da una congiunzione, ad esempio la frase "Luca e Francesca sono al cinema" ha due soggetti legati dalla congiunzione "e".

Modifichiamo la nostra grammatica secondo le considerazioni appena fatte:

S → PR | PR CN S

PR → V | SG V | SG V C

SG → E

V → mangia

C → P E | E

E → A N | A N CN E

A → il | la | i | gli | le | un | una

$N \rightarrow$ gatto | topo

$P \rightarrow$ di | a | da | in | con | su | per | tra | fra

$CN \rightarrow$ e | o | ,

Si noti lo scopo S definito da una o più produzioni in modo ricorsivo e l'analoga struttura per la produzione E. Percorrendo attentamente le produzioni si può notare un'ambiguità nel linguaggio generato, in particolare la produzione S e la produzione E, essendoci la possibilità che abbiano un prefisso comune (SG), richiedono un maggiore sforzo per essere riconosciute. Ad esempio la frase "Elisa è con Luigi e Marco è con Matteo", essendoci la congiunzione "e", è formata da due proposizioni coordinate, ma questo lo si deduce solo dopo che si incontra il secondo predicato "è", infatti la frase "Elisa è con Luigi e Marco" è altrettanto lecita, ma ha una struttura sintattica e logica completamente differente. Il problema risiede nella congiunzione "e" che in un caso lega due proposizioni e nell'altro due entità e per sapere in che caso ci troviamo è necessario fare look-ahead, ovvero leggere avanti un numero imprecisato di volte.

Le problematiche appena esposte sono solo una parte della complessità del linguaggio naturale e riguardano unicamente la parte sintattica. Se volessimo, ad esempio, distinguere i vari tipi di complemento, le preposizioni che precedono i complementi non basterebbero, infatti una preposizione come "con" può introdurre un complemento di mezzo o di compagnia, l'unico modo per capirlo è studiandone la semantica. Si rifletta inoltre sul fatto che una frase può essere corretta dal punto di vista grammaticale, ma assolutamente sbagliata a livello semantico, ad esempio la frase "Un cestino compra una pavimento" nonostante segua le regole grammaticale risulta essere completamente senza senso.

Concluso questo capitolo sul linguaggio che non pretende di essere esaustivo a riguardo, si può passare allo studio degli strumenti disponibili per l'elaborazione del testo.

1.2 Strumenti per l'analisi sintattica del linguaggio naturale

Gli strumenti per l'analisi del linguaggio naturale si possono distinguere in due categorie: quelli mirati all'analisi sintattica e quelli orientati all'analisi semantica. Data la complessità del linguaggio naturale, la distinzione tre le due analisi non è sempre netta, in particolare molto spesso i primi devono tenere in considerazione la semantica per poter fare una buona analisi sintattica. Di seguito vedremo quelli sintattici e successivamente quelli semantici.

Prima di vedere gli strumenti disponibili per l'analisi sintattica è bene evidenziare alcuni concetti impliciti nella teoria del linguaggio esposta nel capitolo precedente.

Quando parliamo di linguaggio naturale, l'alfabeto definito sopra come insieme di simboli non corrisponde all'alfabeto convenzionale, piuttosto coincide col significato comune di vocabolario. Vien da sé che ogni lingua ha il suo vocabolario e la sua grammatica. Per di più i vocaboli si manifestano spesso declinati in base al genere e al numero. Ne consegue che non può esistere uno strumento universale per tutte le lingue, o meglio, va sempre specificata la lingua del testo da analizzare.

Un'ultima considerazione va fatta sulla grammatica: nonostante ogni lingua abbia la sua grammatica, si possono trovare dei punti in comune tra diverse lingue, ad esempio, tutte avranno un soggetto che compie un'azione, l'azione stessa espressa da un verbo ed eventualmente un complemento che specifica qualche dettaglio sull'azione o sul soggetto.

Quasi tutti gli strumenti per NLP condividono i seguenti passaggi nell'elaborazione del testo:

1. separazione del testo in frasi;
2. separazione della frase in token;
3. classificazione dei token tramite tag;
4. costruzione di un albero sintattico;

5. risoluzione di riferimenti (ad esempio i pronomi).

Tralasciano i primi due passaggi, relativamente banali, andiamo a discutere il terzo punto. Le librerie per NLP aggiungono un tag ad ogni parola, o più formalmente, ad ogni simbolo (token) della frase. Il tag corrisponde al ruolo grammaticale della parola, ovvero indica se la parola è un nome, un articolo, un aggettivo, un verbo etc. . A seconda del software utilizzato questi tag possono essere più o meno ricchi di informazioni, per esempio possono dire se un nome è un nome proprio o comune, se è singolare o plurale e via dicendo. Uno dei dizionari di tag più utilizzati è Penn Treebank Tags[1].

Dal tagging è possibile collegarsi direttamente a quanto detto precedentemente sulla grammatica dei linguaggi, infatti i tag identificano, di fatto, quei simboli non terminali della grammatica che producono direttamente simboli terminali, si ricordi N $\rightarrow$ gatto | topo che corrisponde grammaticalmente al concetto di sostantivo o A $\rightarrow$ il | la | i | gli | le | un | una che invece coincide con l'articolo. A questo punto, definita una grammatica, solo tramite l'analisi dei tag, a prescindere dalle parole, è possibile procedere con un'analisi sintattica del testo. Il punto 4 consiste nella costruzione dell'albero sintattico. Ogni software costruisce l'albero in modo leggermente differente. Gran parte delle librerie comunque si basa sulla grammatica inglese, leggermente differente dalla classica analisi italiana "soggetto verbo complemento".

L'ultimo punto è un potente strumento che ha lo scopo di indicare i riferimenti ad un'entità. Questo risolutore di riferimenti è in grado, dato un pronome o un appellativo, di specificare il nome, nel testo, dell'entità a cui il pronome o l'appellativo è riferito.

1.2.1 Apache OpenNLP e Stanford CoreNLP

Per questo lavoro di tesi sono state sperimentate principalmente due librerie per NLP: OpenNLP[13] e Stanford CoreNLP[14].

[1] https://www.cis.upenn.edu/~treebank/

OpenNLP è una libreria open source sviluppata in Java da Apache, che svolge tutte le operazioni descritte precedentemente. Il software è utilizzabile con più lingue, infatti, grazie all'uso di tecniche di machine learning, è possibile addestrarlo con una lingua a piacere, ottenendo così il modello della lingua. Sul sito di Apache OpenNLP sono già disponibili al download diversi modelli per le lingue più diffuse.

OpenNLP è risultato essere molto rapido nell'esecuzione, poco pesante a livello di memoria occupata, ma ha mostrato alcuni limiti critici come l'assenza, nonostante sia scritto il contrario nel sito web, della corefence resolution ed un tagging più limitato rispetto alla prossima libreria che andiamo a presentare.

Stanford CoreNLP è una libreria open source sviluppata, anche questa in Java, dall'università di Stanford. La libreria offre molte funzionalità oltre a quelle viste precedentemente, è possibile usarla con più lingue, ma è particolarmente specializzata nell'inglese. Computazionalmente è molto più pesante di OpenNLP, ma in compenso offre funzionalità più potenti relativamente ai token e alla coreference resolution.

Per quanto riguarda i token, Stanford CoreNLP, oltre a fornire il tag con le informazioni di base, consente di ottenere altre informazioni come il numero (singolare, plurale, sconosciuto), se la parola si riferisce a qualcosa di animato o no e il valore numerico di un numero scritto in lettere, cosa fondamentale per lo scopo della tesi.

1.3 Strumenti per l'analisi semantica

In questo capitolo vengono presentati due strumenti per l'analisi semantica: Il Wikizionario[15] e WordNet[17]. Il primo può fungere anche da analizzatore grammaticale mentre il secondo è uno strumento puramente semantico.

Lo scopo dell'analisi semantica è quello di mettere in relazione le parole al loro significato. Tipicamente ciò viene fatto tramite l'organizzazione, la definizione e la descrizione dei concetti espressi dai vocaboli. L'or-

ganizzazione semantica del lessico si avvale di raggruppamenti messi in relazione semantica. Le relazioni possono essere di diverso tipo:

- Sinonimia: quando due termini si riferiscono allo stesso significato
- Iperonimia: quando una parola esprime un concetto più generico rispetto ad un'altra
- Iponimia: quando un vocabolo ha un significato più specifico di un altro
- Olonimia: se una parola rappresenta una parte di qualcos'altro (ad esempio zampa e animale)
- Meronimia: se un termine esprime qualcosa di cui l'altro termine è una parte (ad esempio animale e zampa)
- Implicazione: quando fare l'azione di un verbo implica farne anche un'altra

L'uso di questi strumenti permette di mettere in relazione concetti espressi in modo differente. Per l'analisi e la comprensione del testo è essenziale riuscire a collegare termini diversi ad un unico concetto comune. È possibile ad esempio dedurre che una macchina ed un camion sono dei veicoli o che *possedere* un oggetto è la stessa cosa di *avere* un oggetto.

1.3.1 Wikizionario e JWKTL

Il Wikizionario è un progetto collaborativo supportato da Wikimedia Foundation[2], multilingue e gratuito con l'obiettivo di realizzare un dizionario online con significati, etimologie, pronunce e reti semantiche. Lo scopo finale è quello di raccogliere e classificare tutti vocaboli di tutte le lingue note. Essendo un progetto collaborativo, questo dizionario, è disponibile in ben 172 lingue ed è in continua crescita.

Data una parola, il Wikizionario fornisce le sue forme alterate, per esempio la forma dialettale, la sua pronuncia, anche in formato audio,

[2]https://wikimediafoundation.org

l'etimologia, tutti i ruoli grammaticali che può ricoprire con relativi significati ed esempi d'uso, concetti correlati secondo le relazioni semantiche descritte sopra e traduzioni e relazioni rispetto ad altre lingue.

Il dizionario è disponibile online tramite un'interfaccia web, ma fornisce anche un archivio strutturato scaricabile e manipolabile tramite un'interfaccia Java, in particolare per mezzo della libreria JWKTL[16] (Java Wiktionary Library) libera ed opensource.

1.3.2 WordNet

WordNet è un database lessico-semantico sviluppato dall'Università di Princeton per la lingua inglese. Il suo scopo è quello di organizzare i vocaboli secondo raggruppamenti chiamati synset messi in relazione semantica fra loro. Il synset è definito come l'insieme dei vocaboli che esprimono lo stesso concetto, mentre le relazioni tra di essi sono quelle elencate sopra ovvero quelle di iperonimia, iponimia, olonimia, meronimia e implicazione. Il database conta circa 155 mila parole organizzate in 117 mila synset. WordNet, però, non include tutte le parole della lingua inglese, in particolare considera solo i nomi, i verbi, gli aggettivi e gli avverbi, ignora invece le preposizioni, gli articoli e le altre categorie grammaticali.

Il sistema deve essere interrogato specificando la parola cercata, la categoria grammaticale e il tipo di relazione semantica di interesse, come risposta, WordNet riporta tutti i synset di cui fa parte quella parola intesa nel ruolo grammaticale specificato e tutti i synset collegati tramite la relazione semantica specificata. A differenza del Wikizionario, WordNet fornisce solo le relazioni semantiche tra i vocaboli e una breve glossa per ogni termine.

Il database è disponibile liberamente tramite web, librerie Java ed è anche integrabile in Prolog[1].

1.4 Risoluzione di problemi matematici

In questa sezione verranno discussi a titolo di esempio alcuni progetti di ricerca riguardanti la risoluzione di problemi matematici descritti in linguaggio naturale. Poiché questi progetti condividono lo stesso obiettivo di questa tesi, è interessante confrontarli per capire quali siano i problemi matematici che intendono risolvere, le metodologie utilizzate e i risultati ottenuti. Dall'analisi di questi lavori è emerso che gli approcci usati si possono distinguere sostanzialmente in due macro categorie: simbolico e statistico basato su apprendimento.

Il primo, quello simbolico, è caratterizzato da un'analisi precisa del testo, viene processato tutto il documento, parola per parola e, da quell'analisi, se ne ricava in modo deterministico un accurato modello matematico. Questa metodologia porta a risultati esatti poiché ogni interpretazione di ogni parte della frase è scritta manualmente. Strumenti come WordNet aiutano molto a ridurre la mole di codice da scrivere, infatti, data l'interpretazione di un concetto, WordNet riduce le sue diverse rappresentazioni a quel unico concetto permettendo quindi di associare la rispettiva interpretazione. Ad esempio, se si volesse implementare l'interpretazione del verbo "avere", basterebbe associarla al concetto di "avere" e tramite WordNet la stessa implementazione potrebbe essere utilizzata per tutti i sinonimi di "avere", come "possedere", "detenere" etc., forniti automaticamente dallo strumento semantico.

Nonostante i diversi accorgimenti che si possono attuare, dare semantica ad ogni possibile testo in questo modo rende il lavoro impossibile. Perché allora usare questo approccio? È ragionevole pensare che, concentrandosi su testi di problemi matematici, il campo della semantica, quindi delle interpretazioni, si riduca di molto. Si pensi che, oltre a quanto appena detto, i testi matematici sono solitamente più formali e, nei quesiti soprattutto, si cerca il più possibile di ridurre le ambiguità rendendo più agevole l'individuazione di un unico significato. È inoltre consuetudine calare problemi matematici in situazioni e contesti quotidiani, ma la comprensione di tali testi implica di fatto una serie di conoscenze e ca-

pacità deduttive che riportano il compito ad un livello di difficoltà quasi pari alla comprensione di un generico testo. Nei progetti che verranno descritti nel seguito sono state fatte delle ulteriori scelte di riduzione delle interpretazioni e di conseguenza dei problemi risolubili.

Il secondo approccio, quello statistico, fa solitamente uso di tecniche di apprendimento automatico e crea il modello matematico usando modelli o parti di modelli apprese dal *training set*, ovvero un insieme di esempi con relative soluzioni. La realizzazione di questo metodo prevede, prima di poter usare il software, una fase di addestramento. Viene costruito un *training set* composto da un adeguato numero di problemi, possibilmente equilibrato secondo la tipologia, per ognuno si esegue un'operazione di annotazione e si associa il modello matematico corrispondente e/o la soluzione. La creazione del *training set* viene solitamente fatta, almeno in parte, a mano poiché è difficile trovare tale materiale già pronto con le proprietà desiderate. Una volta realizzato, il *training set* viene dato in input all'algoritmo di apprendimento automatico che ne ricava un suo modello interno che varrà poi usato per l'elaborazione e risoluzione dei quesiti. La precisione di tali applicazioni dopo l'addestramento dipende da diversi fattori come l'algoritmo di addestramento utilizzato e la composizione dell'insieme di esempi in quanto a numero di esempi, bilanciamento del tipo di esempi e metodo di annotazione.

Sebbene possa non sembrare una tecnica rigorosa come richiederebbe un metodo matematico, questo criterio è simile a quello che anche le persone usano nella pratica: leggendo il testo si intuisce che la tipologia del problema è simile ad un'altra già incontrata, quindi si sa quali sono i modelli matematici candidati alla risoluzione e, da lì in poi, si ragiona come meglio adattare quei modelli al problema in esame.

Come si potrà notare dai sistemi software realizzati secondo quest'ultimo approccio, il risultato finale è più scarso in termini di correttezza, ma è migliore per quanto riguarda il numero di problemi che riesce a comprendere. Infatti testi più complessi a livello sintattico/semantico risultano essere più facilmente interpretabili con il metodo statistico rispetto a quello simbolico.

1.4.1 Applicazioni per la risoluzione di problemi matematici

In questa sezione verranno presentati alcuni lavori esistenti che risolvono problemi matematici descritti in linguaggio naturale. Alcuni di questi progetti adottano un approccio simbolico, altri statistico e altri ancora una unione dei due.

1.4.2 SigmaDolphin

SigmaDolphin[2] è il nome del sistema sviluppato dalla University of Science and Technology of China e fa uso di tecniche basate sull'analisi semantica e ragionamento. L'architettura del software è organizzata in tre macro passi: creazione di un modello intermedio dal testo, costruzione del modello matematico da quello intermedio e, infine, risoluzione.

Il primo passaggio esegue un'analisi semantica del testo volta a costruire un modello che verrà poi rielaborato. Questo modello intermedio è implementato da un linguaggio ad hoc chiamato DOL (Dolphin Language) creato da loro stessi appositamente. Questa rappresentazione intermedia cattura il significato del testo e presenta una struttura più facile da elaborare. Problemi come i diversi riferimenti alle stesse entità possono essere facilmente risolti navigando l'albero di questo linguaggio. Il testo viene analizzato come si fa per un linguaggio libero da contesto, SigmaDolphin fa uso di oltre 9600 regole grammaticali per costruire l'albero DOL.

La seconda fase, implementata dal modulo di ragionamento, elabora il modello intermedio per creare il modello matematico. Durante questa fase vengono applicati algoritmi di iperonimia per comprendere meglio le relazioni tra le entità quindi vengono create le equazioni del problema. Infine vengono risolte le equazioni che restituiranno il risultato finale.

L'insieme dei problemi che è in grado di risolvere è limitato ai soli problemi descritti in termini matematici ovvero non è in grado di dedurre un modello matematico da un problema calato, ad esempio, in un con-

testo di vita reale poiché questo implicherebbe, come già spiegato precedentemente, una conoscenza di base e delle capacità di relazione piuttosto complesse. Un esempio di problema che è in grado di risolvere è il seguente:

Es. 1.4.1. "One number is 16 more than another. If the smaller number is subtracted from 2/3 of the larger, the result is 1/4 of the sum of the two numbers. Find the numbers."

SigmaDolphin ricade nella categoria dei metodi simbolici poiché sia per la parte dell'analisi del testo che nella parte di ragionamento vengono utilizzate tecniche di elaborazione e di rappresentazione tipiche di questa categoria. SigmaDolphin ha una precisione (numero di risposte corrette su numero di quesiti a cui viene data la risposta) del 95,4% con una copertura (numero di risposte corrette su numero di quesiti totali) pari al 60,2%. Il sistema, quando dà la risposta, ha un'alta probabilità che questa sia giusta, in più è interessante sapere che quando il sistema non riesce a rispondere, per la gran parte dei casi, la causa è una mancata implementazione di una particolare parte di interpretazione del testo.

1.4.3 KAZB

KAZB[3] è un sistema elaborato dall'Università di Washington basato su tecniche di apprendimento automatico e template (modelli predefiniti creati a priori). Il software, come nel caso precedente, è in grado di risolvere problemi matematici che non necessitano di una conoscenza di base troppo complessa e che sono risolvibili tramite l'algebra lineare.

Es. 1.4.2. "An amusement park sells 2 kinds of tickets. Tickets for children cost $ 1.50. Adult tickets cost $ 4 . On a certain day, 278 people entered the park. On that same day the admission fees collected totaled $ 792 . How many children were admitted on that day? How many adults were admitted?"

Essendo basato su tecniche di apprendimento per prima cosa il programma è stato addestrato tramite un *training set* di problemi matematici

annotati con i relativi sistemi di equazioni e le rispettive soluzioni. In questo modo il sistema impara, con un certo livello di confidenza, ad associare ad un quesito un modello predefinito di equazioni. Si parla di template perché le equazioni predefinite disponibili hanno i coefficienti non definiti. Una volta scelto il modello viene fatta un'analisi del testo per associare le entità, in particolare le quantità delle entità, ai coefficienti delle equazioni.

Dai test effettuati risulta che, su un database di 514 problemi matematici presi da Algebra.com, il sistema sopra descritto è in grado di risolvere con successo una percentuale pari al 70%.

1.4.4 ARIS

Il prossimo sistema preso in esame è ARIS[4] ed è stato sviluppato da Allen Institute for AI. A differenza dei progetti visti precedentemente, ARIS adotta un approccio misto nella risoluzione dei problemi e si focalizza in particolar modo sulla classificazione dei verbi.

Il modello su cui si basa è definito da entità e rispettivi attributi, contenitori e transizioni di stato. Il concetto dominante è quello di osservare i cambiamenti delle entità attuati tramite i verbi. I verbi costituiscono un ruolo fondamentale nella relazione tra le entità. Questi vengono classificati in sette categorie e per ognuna è definita un'operazione matematica di somme e/o sottrazioni che andranno a modificare lo stato delle entità. Lo stato finale, o comunque uno degli stati a seconda della domanda del problema, conterrà la soluzione.

Tecniche di apprendimento vengono usate per classificare i verbi: è stata addestrata una *Support Vector Machine*[3] per catalogare i verbi secondo tre diverse caratteristiche, una di queste si basa sul sopra citato WordNet. Il sistema risulta molto robusto e riesce a risolvere il 77% del test set costituito da circa 400 problemi matematici.

Un esempio di problema che ARIS è in grado di risolvere è il seguente:

[3]Si tratta di un classificatore che fa uso di un algoritmo di addestramento automatico supervisionato.

Es. 1.4.3. "Liz had 9 black kittens. She gave some of her kittens to Joan. Joan now has 11 kittens. Liz has 5 kittens left and 3 have spots. How many kittens did Joan get?"

1.5 Conclusioni

In questo primo capitolo sono state discusse le basi per l'analisi del linguaggio naturale e le maggiori problematiche correlate ad essa, sono stati presentati alcuni strumenti disponibili per l'elaborazione dei testi ed infine esaminati alcuni progetti di ricerca finalizzati alla risoluzione di problemi matematici descritti in linguaggio naturale.

Da questa analisi si deduce che le possibilità di approccio al problema sono diverse: da una parte mezzi analitici, deterministici che richiedono un maggiore sforzo implementativo, ma presentano una maggiore precisione nel risultato e dall'altra parte tecniche di apprendimento automatico più approssimative nei risultati, ma più flessibili e robuste nell'analisi. Ovviamente è possibile seguire un approccio misto per cercare di sfruttare al meglio i vantaggi delle due tecniche e colmarne i difetti.

Nel prossimo capitolo si parlerà dell'obiettivo di questo lavoro di tesi, quali tipi di problemi si intende risolvere e quali sono i concetti chiave che hanno guidato il progetto.

Capitolo 2

Descrizione del problema e principi realizzativi

In questo lavoro di tesi si vogliono risolvere in modo automatico problemi matematici descritti in linguaggio naturale. L'insieme dei problemi è circoscritto a quesiti di tipo logico-matematico e aritmetico espressi in lingua inglese.

Non vengono posti limiti sulla descrizione del problema, ovvero la descrizione può essere diretta, quindi espressa in termini puramente matematici, oppure calata in un contesto di vita reale. Un esempio di problema nella prima forma è il seguente:

Es. 2.0.1. "La somma di due numeri è 85. La differenza tra il doppio del primo e il secondo è 5. Trovare i due numeri."

Mentre un esempio nella seconda forma può essere:

Es. 2.0.2. "Dieci amici organizzano una gara di tiro alla fune. I pettorali sono numerati da 1 a 10. I partecipanti si dividono in due squadre: i Potenti e gli Smilzi. Il prodotto dei pettorali dei componenti della squadra dei Potenti è un numero uguale alla somma dei pettorali dei componenti della squadra degli Smilzi. Qual è questo numero?"

Ciò che viene richiesto dal primo quesito è esplicito e per risolverlo è necessario conoscere i concetti matematici presenti nel testo. Nel secondo problema, invece, il modello matematico è celato dietro la descrizione

di una situazione reale, in questo caso sportiva, ciò implica che, oltre alla necessità di conoscere i concetti matematici, è necessario comprendere le relazioni esplicite e implicite tra le entità presenti e trovarne una corrispondenza matematica. Per fare questo, oltre a dover fare un'analisi semantica più complessa, è necessario possedere una conoscenza di base. In questo caso particolare è indispensabile sapere che i pettorali sono univoci e che ogni partecipante ha il suo pettorale. La relazione pettorale-partecipante non è ricavabile al testo, ma è una nozione che bisogna avere a priori.

La comprensione del testo si traduce nella deduzione di un modello matematico capace ti risolvere il problema. Questo modello può essere rappresentato, ad esempio, tramite una rappresentazione algebrica costituita da un sistema di equazioni. Il quesito 2.0.2 è descritto dalle seguenti equazioni:

$$\prod_{i=1}^{Np} x_i = \sum_{i=Np+1}^{Ns} x_i, x_i \in [1,10],$$

$$Np + Ns = 10, Np \in [1,9], Ns \in [1,9],$$

$$x_1 \neq x_2, x_1 \neq x_3, x_1 \neq x_4, x_1 \neq x_5, x_1 \neq x_6, x_1 \neq x_7, x_1 \neq x_8, x_1 \neq x_9, x_1 \neq x_{10},$$

$$x_2 \neq x_3, x_2 \neq x_4, x_2 \neq x_5, x_2 \neq x_6, x_2 \neq x_7, x_2 \neq x_8, x_2 \neq x_9, x_2 \neq x_{10},$$

$$x_3 \neq x_4, x_3 \neq x_5, x_3 \neq x_6, x_3 \neq x_7, x_3 \neq x_8, x_3 \neq x_9, x_3 \neq x_{10},$$

$$x_4 \neq x_5, x_4 \neq x_6, x_4 \neq x_7, x_4 \neq x_8, x_4 \neq x_9, x_4 \neq x_{10},$$

$$x_5 \neq x_6, x_5 \neq x_7, x_5 \neq x_8, x_5 \neq x_9, x_5 \neq x_{10},$$

$$x_6 \neq x_7, x_6 \neq x_8, x_6 \neq x_9, x_6 \neq x_{10},$$

$$x_7 \neq x_8, x_7 \neq x_9, x_7 \neq x_{10},$$

$$x_8 \neq x_9, x_8 \neq x_{10},$$

$$x_9 \neq x_{10} \tag{2.1}$$

La soluzione del modello sopra descritto non può essere trovata semplicemente eseguendo i calcoli, ma è necessario procedere provando tutti gli assegnamenti possibili fino a trovare quello che soddisfa tutte le equazioni. Un problema di questo tipo può essere visto come un problema di soddisfacimento di vincoli.

Un problema di soddisfacimento di vincoli è definito su un insieme finito di variabili i cui valori appartengono a domini finiti e su un insieme di vincoli. Per vincoli si intendono quelli base della matematica come l'equazione, il maggiore, maggiore uguale, minore, minore uguale, disuguaglianza e via dicendo più vincoli maggiormente complessi come il vincolo

di *allDifferent*. L'insieme delle disequazioni dell'esempio sopra rappresenta il vincolo di *allDifferent*, ovvero impone che tutte le variabili devono avere valori differenti tra loro. In un linguaggio di programmazione logica questo può essere espresso tramite il predicato *allDifferent(L)* dove *L* è la lista delle variabili che devono soddisfare il vincolo. Il modello suddetto può essere riscritto in linguaggio logico come segue:

$$
\begin{aligned}
tugOfWar(Min, Max, L1, L2, Z) :- \\
&numberList(Min, Max, Numbers), \\
&allDifferent(Numbers), \\
&permutation(Permutation, Numbers), \\
&append(L1, L2, Permutation), \\
&productList(L1, Z), sumList(L2, Z).
\end{aligned}
$$

dove *Min* e *Max* equivalgono rispettivamente a 1 e 10, *L1* e *L2* rappresentano i pettorali della squadra dei Potenti e degli Smilzi e *Z* è la soluzione richiesta dal problema ovvero la produttoria dei pettorali dei Potenti e la sommatoria dei pettorali degli Smilzi. In questo caso il vincolo di *allDifferent* è ridondante e può essere omesso perché è implicito nel predicato *numberList* che genera tutti i numeri interi ordinati da *Min* a *Max* compresi.

La programmazione logica inoltre non costringe i domini delle variabili a valori numerici, infatti, poiché questo tipo di programmazione lavora sui simboli, il dominio può essere definito anche da stringhe, ciò permette di modellare più facilmente altri problemi come il seguente:

Es. 2.0.3. "Trovare il più piccolo numero che, aggiunto alla somma delle sue cifre, è uguale a 2011."

In questo caso "il più piccolo numero" va interpretato prima come stringa, per ottenere le cifre, e poi come numero per fare la somma.

Una volta ottenuto il modello in una qualche forma è possibile ottenere la soluzione tramite un risolutore automatico capace di interpretare quella determinata rappresentazione del modello. Il lavoro di questa tesi, quindi, si concentra sui passaggi necessari a dedurre dal testo in linguaggio naturale del quesito matematico un modello matematico risolvibile da un risolutore automatico.

Nella prossima sezione andiamo a vedere quali sono i passaggi necessari per ricavare il modello matematico e con quali metodologie si è deciso di implementarli.

2.1 L'architettura generale

L'architettura generale del sistema prevede tre macro fasi:

1. costruzione di un modello di scopo generale del testo;
2. analisi del modello precedente mirata alla deduzione del modello matematico;
3. risoluzione del modello matematico tramite un risolutore automatico.

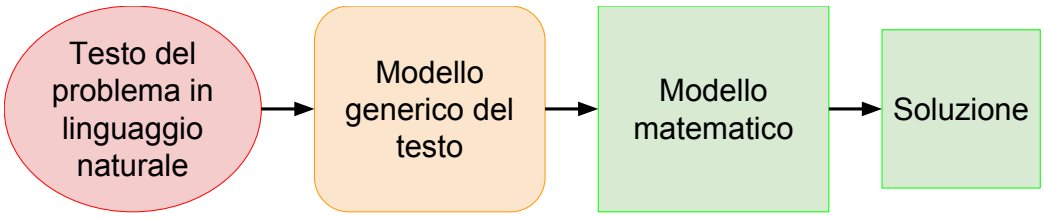


Figura 2.1: L'architettura generale

Poiché i testi dei quesiti matematici tendono ad essere formali, sintatticamente poco articolati e con meno ambiguità possibili, l'approccio scelto per per tutte le fasi è di tipo puramente simbolico. Nel seguito andiamo a descrivere in modo più approfondito i tre punti appena menzionati.

2.1.1 Analisi del testo

Il primo punto consiste nell'analisi grammatica e logica del testo. La prima è svolta in gran parte da un *framework* esistente che fornisce, come descritto nel capitolo 1.2, informazioni come il ruolo grammaticale (nome, aggettivo, etc.), il genere e il numero per ogni parola. L'analisi logica è realizzata quasi interamente da un modulo sviluppato in questo lavoro di tesi e si preoccupa di individuare le parti logiche delle frasi quali proposizioni, soggetto, predicato verbale e complementi. Questa prima fase non tiene conto di nessun concetto matematico, piuttosto mantiene un'ottica generale e risolve problematiche relative al linguaggio naturale come l'individuazione delle proposizioni, la risoluzione di pronomi e soggetti sottintesi. L'unica eccezione riguarda l'individuazione e la modellazione precisa delle quantità relative alle entità espresse nel testo.

Il prodotto di questa analisi consta in un modello interno, nello specifico scritto in linguaggio Java, che cattura la struttura logica del testo nella quale i riferimenti impliciti, quali pronomi e soggetti sottintesi, sono resi in modo esplicito.

2.1.2 Deduzione del modello matematico

La fase successiva è la più complessa e costituisce il cuore del lavoro di questa tesi. Dalla rappresentazione del testo ricavata nel punto precedente si costruisce il modello matematico. Studiando la logica delle frasi del linguaggio naturale si evince che la parte logica più importante è il verbo poiché è quest'ultimo che dà significato alle proposizioni mettendo in relazione le entità menzionate nel testo. Quindi le relazioni esplicite tra le entità sono ricavabili dallo studio del predicato verbale che le associa secondo una certa semantica. Poiché l'intento di questo lavoro di tesi è mirato ad una traduzione matematica del testo, il problema si può ridurre nell'attribuire un significato matematico ai verbi, trasformando questi in equazioni o vincoli che leghino le entità coinvolte. Per esempio la frase "Luigi ha cinque automobili" può essere tradotta in termini di equazioni

come segue:

$$automobile_{luigi} = 5 \tag{2.2}$$

L'equazione è molto semplice e significa che la variabile che rappresenta le automobili di Luigi $automobile_{luigi}$ ha valore 5. In generale quindi si traduce l'entità "automobili di Luigi" nella rispettiva variabile ed il verbo "avere" si riflette nell'equazione che assegna un valore a questa variabile.

Una rappresentazione in linguaggio logico, invece, potrebbe essere la seguente:

$$avere(5, automobile, luigi). \tag{2.3}$$

Questo fatto logico può essere generalizzato nella forma *avere(X,Oggetto,Soggetto)* interpretato come: il "Soggetto" (Luigi) "avere" (ha) "X" (5) "Oggetto" (automobili).

L'esempio del verbo avere è tra i più semplici, nei capitoli successivi si vedranno nel dettaglio quali interpretazioni di verbi sono state implementate ed in che modo. Naturalmente non è possibile pensare che l'implementazione di queste traduzioni sia stata fatta verbo per verbo: sono state individuate diverse categorie di predicati verbali alle quali si è associata una semantica matematica. Per capire in quale categoria ricade ogni verbo si è fatto uso del concetto di iperonimia tramite strumenti semantici quale WordNet. Riprendendo l'ultimo esempio, se la frase fosse stata "Luigi **possiede** cinque automobili" il risultato sarebbe stato lo stesso poiché il significato di possedere equivale a quello di avere.

L'obiettivo finale di questa fase è quello di ottenere un modello matematico del problema. Come appena visto, il modello può essere rappresentato in forma algebrica o in forma logica. In questo lavoro sono state sperimentate entrambe le rappresentazioni, in particolare si sono distinte due tipologie di problemi: quelli logico-matematici e quelli aritmetici, per ognuna delle tipologie è stato implementato un modulo specifico in grado di ricavarne il rispettivo modello matematico.

Introduciamo ora un altro concetto necessario per risolvere alcune tipologie di esercizi e che fa parte del modello che si vuole realizzare: lo stato. Alcuni quesiti descrivono una situazione statica e richiedono di ri-

cavare il valore di una qualche incognita, altri quesiti, invece, descrivono una situazione che muta nel tempo e chiedono il valore di una qualche variabile in un determinato momento. Prendiamo ad esempio il seguente problema:

Es. 2.1.1. "Giacomo ha 400 euro, Lucia ha 600 euro e Francesco ha 320 euro. Giacomo dà 200 euro a Lucia. Lucia dà 25 dollari a Francesco. Quanti soldi ha Lucia?"

Nella prima frase viene descritto lo stato iniziale che, coerentemente al modello algebrico, è rappresentato dalle tre variabili $euro_{giacomo}$, $euro_{lucia}$ e $euro_{francesco}$ di valore rispettivamente 400, 600 e 320. Nelle frasi successive le stesse tre variabili dovranno cambiare valore, in particolare dalla seconda frase si deduce che Giacomo avrà 200 euro in meno mentre Lucia ne avrà 200 in più, ragionamento equivalente per la frase successiva. Il problema chiede il valore della variabile $euro_{lucia}$ alla fine delle transizioni.

Illustrati i concetti e la struttura del modello matematico, nel capitolo successivo si vedrà la terza macro fase nella quale questo modello viene rielaborato e successivamente risolto dando la soluzione del problema.

2.1.3 Risoluzione del modello matematico

In quest'ultima fase il modello matematico è quasi pronto per essere risolto, prima è necessario controllare se questo è coerente ed, in caso, renderlo tale. I risolutori automatici sono sistemi formali che hanno bisogno di coerenza e precisione altrimenti falliscono l'esecuzione. L'incoerenza del modello può scaturire dal fatto che alcune variabili con nomi diversi in realtà stiano ad indicare la medesima cosa. Nell'esempio 2.1.1 abbiamo dato per scontato che il problema richieda il valore della variabile $euro_{lucia}$, ma seguendo attentamente la logica della seconda fase, dalla domanda "Quanti soldi ha Lucia?" l'algoritmo dovrebbe generare la variabile incognita col nome $soldo_{lucia}$ e non $euro_{lucia}$, infatti la fase precedente genera proprio $soldo_{lucia}$. La trasformazione in $euro_{lucia}$ viene fatta in quest'ultima fase.

Accompagniamo la descrizione di questa parte con il seguente esempio:

Es. 2.1.2. "Giovanni ha 5 macchine. Marco ha 6 trattori. Quanti veicoli hanno in totale?"

Il quesito chiede il numero totale di veicoli, ma il modello generato non ha alcuna variabile con valore noto che riconduca ad un qualche veicolo, infatti il modello generato è il seguente:

$$\begin{aligned} & macchina_{giovanni} = 5, \\ & trattore_{marco} = 6, \\ & veicolo_{giovanni} + veicolo_{marco} = veicolo_{totale} \end{aligned} \tag{2.4}$$

Usando WordNet è stato implementato un algoritmo che, per mezzo dell'iperonimia, sostituisce le variabili incognite $veicolo_{giovanni}$ e $veicolo_{marco}$ con le rispettive $macchina_{giovanni}$ e $trattore_{marco}$. In questo modo il modello diviene coerente e quindi risolubile da un risolutore automatico.

2.2 Conclusioni

In questo capitolo sono state descritte le tipologie di problemi che si intende risolvere e tramite quale approccio. In particolare si è visto che il metodo verte principalmente sull'analisi dei verbi e la loro traduzione in termini matematici. L'elaborazione del testo e del modello matematico è reso poi più flessibile e robusto tramite strumenti semantici come WordNet.

L'approccio alla risoluzione è puramente simbolico: sono state individuate una serie di categorie di verbi ed un insieme di entità e per ognuna di queste è stata scritta una interpretazione matematica.

Come si è detto e come si è potuto notare dagli esempi riportati in questo capitolo, le tipologie di problemi che si vogliono affrontare sono

diverse ed in particolare sono classificabili in problemi logico-matematici (2.0.2) e problemi aritmetici (2.1.2). Allo scopo di poter risolvere al meglio entrambe le tipologie sono stati sviluppati due moduli distinti per la deduzione dei rispettivi modelli.

Nel prossimo capitolo si andrà a descrivere in modo dettagliato la prima parte che è stata implementata e che riguarda il primo punto dell'architettura generale ovvero l'analisi del testo finalizzata alla costruzione di un modello di testo di scopo generale.

Capitolo 3

Implementazione dell'analisi del linguaggio naturale

In questo capitolo verrà descritto in modo dettagliato il lavoro svolto per ottenere, dal testo in linguaggio naturale, una struttura capace di rappresentare al meglio le parti logiche delle frasi. Ciò che si vuole ottenere in questa prima parte è un modello del testo che rispecchi l'analisi logica del periodo ovvero il riconoscimento delle proposizioni e le relazioni fra di esse, l'individuazione dei soggetti, dei verbi e dei complementi. La rappresentazione risultante, oltre a catturare i concetti dell'analisi logica, deve essere in grado di esplicitare più informazioni possibili di modo che una successiva rielaborazione di questo modello possa essere il più agevole possibile. Per questo motivo è necessario affrontare e risolvere alcune problematiche del linguaggio naturale come i riferimenti espressi dai pronomi e i soggetti sottintesi.

Questa parte del lavoro è stata implementata in Java, quindi verranno usati dei grafici UML in ausilio alla descrizione testuale.

3.1 Il modello del testo

Iniziamo col descrivere il modello che vogliamo ottenere. Un documento è un insieme di frasi, ogni frase può essere composta da una o più

proposizioni. Il diagramma seguente illustra questa semplice composizione.

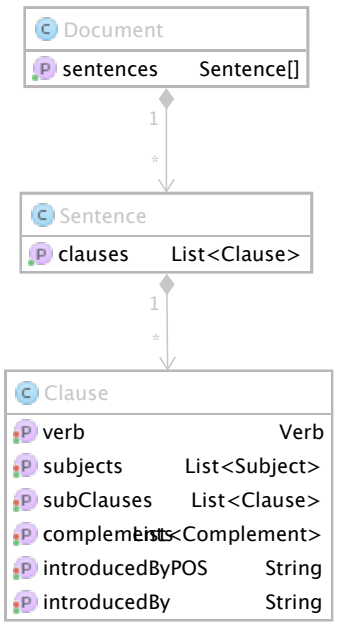


Figura 3.1: Il documento, le frasi e le proposizioni

Le proposizioni possono essere in relazione coordinata o subordinata. La coordinazione è modellata dalle liste *clauses* e *subClauses*, proprietà rispettivamente di *Sentence* e *Clause*: tutte le proposizioni nella stessa lista sono allo stesso livello logico quindi corrispondono alla coordinazione. Le proposizioni subordinate appartengono alla sotto-lista *subClauses* della proposizione da cui dipendono.

Come si può notare ancora dal diagramma sopra, una proposizione è costituita da un verbo, una lista di soggetti e un'altra di complementi. È importante ricordare che nell'analisi logica ad ogni verbo corrisponde una proposizione, da ciò se ne deduce che la proposizione è l'unità con senso compiuto più semplice possibile. L'analisi della proposizione consisterà nello studio del suo verbo e delle entità che esso mette in relazione.

Il modello si espande con le classi *Subject* e *Complement* che non sono nient'altro che sottoclassi di *Entity* come mostrato nel diagramma succes-

sivo:

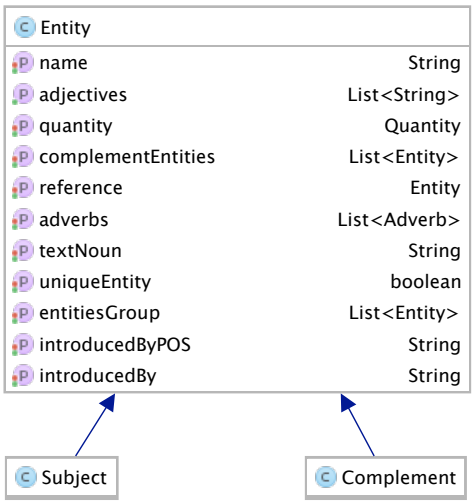


Figura 3.2: La classe Entità

La classe *Entity* rappresenta, appunto, una certa entità e gioca un ruolo fondamentale nell'analisi del testo. È identificata dal suo nome e la sua descrizione è arricchita da aggettivi, complementi e da una quantità. La classe *Quantity* può essere di diversi tipi a seconda della relazione espressa nel testo, per esempio se il testo dice "ci sono meno di 20 partecipanti" la quantità sarà modellata dalla classe *SmallerQuantity* che di fatto esprime il vincolo matematico $partecipanti < 20$, ragionamento equivalente per forme di frase come "... più di ...", "... maggiore o uguale a ..." e via dicendo. La tassonomia completa della classe *Quantity* è descritta dal diagramma seguente:

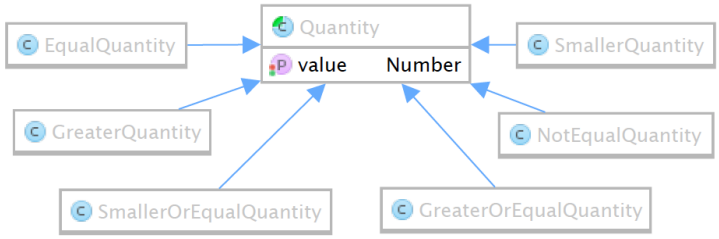


Figura 3.3: La classe Quantity

La classe *Entity* espone anche le proprietà *introducedBy* e *introducedBy-POS*, solitamente presenti solo per i complementi, utili ad identificare il tipo di complemento. Ad esempio la preposizione "of", in Inglese, introduce solitamente un complemento di specificazione; la preposizione "to" può introdurre un complemento di moto a luogo ("I go to the cinema") o un complemento di termine ("I give it to Mario"), in questo caso il tipo del complemento deve essere dedotto non solo dalla preposizione, ma anche dalla semantica.

Un'altra proprietà utile è *entitiesGroup*. Questa proprietà serve a ricordare se l'entità è stata presentata nel testo insieme ad altre, ad esempio "Luca, Paolo e Gianmarco giocano a pallone" i tre nomi saranno nella stessa lista *entitiesGroup*. Questo tornerà utile più avanti nella risoluzione dei pronomi, maggiori dettagli nel capitolo dedicato.

Nel linguaggio naturale è consuetudine fare riferimento ad un certa entità tramite espressioni differenti: si possono usare sinonimi, epiteti, pronomi o intere proposizioni per riferire ad una entità nominata precedentemente. Per catturare questo collegamento tra la citazione nel testo di una entità e l'entità a cui si riferisce viene usata la proprietà *reference* presente nella classe *Entity*. Tale proprietà è di fondamentale importanza per la comprensione del testo e nello specifico in questa tesi sarà utilizzata per la creazione corretta e univoca delle variabili. Si rimanda al capitolo 3.3 tutto il ragionamento che sta dietro a questa proprietà.

Nel prossimo capitolo andiamo a vedere come viene costruito questo modello a partire dall'analisi grammaticale.

3.2 Costruzione del modello

Per ottenere il modello logico del testo è necessario partire dall'analisi grammaticale. In questo lavoro di tesi sono stati sperimentati due strumenti per l'analisi del testo: OpenNLP e Stanford CoreNLP. Per ognuno di questi è stato implementato il rispettivo modulo per la costruzione del modello logico. È doveroso ricordare che l'analisi del linguaggio naturale è un'area di ricerca non ancora matura e che presenta ancora dei problemi aperti e diverse imprecisioni negli strumenti esistenti. I principali problemi attualmente aperti riguardano la comprensione delle relazioni temporali espresse nel testo e la risoluzione dei riferimenti, per di più si presentano difficoltà anche a riconoscere cose apparentemente più semplici come gli aggettivi. Si pensi che il ruolo grammaticale di una parola non è determinato solamente dalla parola stessa, ma dipende anche dal contesto, ad esempio la parola in Inglese "even" è spesso usata come avverbio per significare "anche", "addirittura", ma può essere intesa come aggettivo dal significato "pari" se compare in una frase come "a even number".

I due strumenti sopracitati svolgono diversi compiti: separano il testo in frasi, aggiungono un *tag* ad ogni parola che identifica il ruolo grammaticale e creano una albero sintattico seguendo regole basate sull'analisi inglese dei sintagmi. Una prima implementazione è stata realizzata sull'albero sintattico generato da OpenNLP. Tramite l'uso della ricorsione l'albero viene esplorato e tradotto nel modello di testo interno. Nella realizzazione di questa parte di codice si sono riscontrate due carenze da parte del *framework* utilizzato: mancanza della traduzione dei numeri scritti in lettere in variabili numeriche manipolabili dal calcolatore e assenza di supporto per la risoluzione dei riferimenti. Per questi motivi lo sviluppo basato su OpenNLP è stato interrotto a favore di Stanford CoreNLP.

Anche della seconda libreria è stato sfruttato l'albero sintattico generato per ricavare il modello di testo. A parte piccole differenze, gli alberi generati dai due framework sono uguali quindi è stato possibile riutilizzare parte del codice già scritto. La traduzione dei numeri espressi sotto forma di stringa è supportata e funziona perfettamente, questa era indispensabi-

le per la creazione delle istanze di *Quantity*. Purtroppo, con l'aumentare della complessità della frase, l'albero generato non era sempre preciso e generava una struttura non corretta. Proposizioni coordinate troppo lunghe, molteplici subordinate nonché complementi a seguito di soggetti generavano un albero o scorretto o troppo complesso da essere esplorato. Date queste difficoltà si è deciso di implementare un ulteriore modulo che non facesse uso dell'albero sintattico.

Il terzo modulo implementato usa solo le frasi con i *tag* dell'analisi grammaticale e costruisce direttamente il modello logico del testo. Il modello logico del testo non è nient'altro che l'albero sintattico rispetto le regole dell'analisi logica italiana. I due moduli precedenti, di fatto, traducevano un albero sintattico basato sull'analisi dei sintagmi in un albero basato sull'analisi logica. L'implementazione di questo modulo, invece, usa solo e direttamente le foglie, ovvero le parole arricchite con i *tag*, per costruire l'albero. Il codice consta in una classe con un metodo specifico per ogni etichetta più diverse strutture dati di tipo *stack* per gestire preposizioni, congiunzioni e proposizioni annidate. Per evitare una lunga lista di "if", uno per ogni etichetta, i metodi hanno lo stesso nome delle etichette e vengono chiamati tramite *reflection*.

La tassonomia finale dei moduli implementati è la seguente:

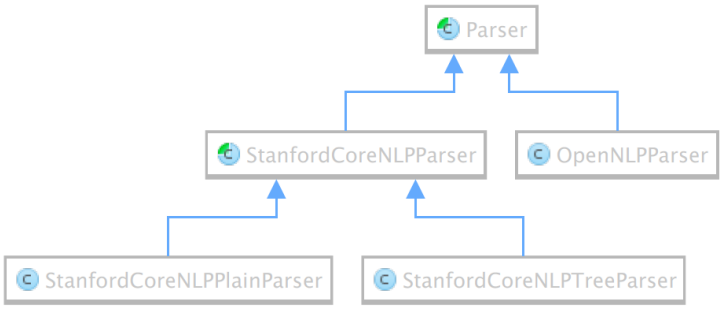


Figura 3.4: Tassonomia dei parser

Poiché tutti i *parser* derivano direttamente o indirettamente da un'unica classe astratta *Parser*, essi condividono i metodi principali per la conversione del testo dalla forma di stringa o derivante da un generico flusso di

input a *Document*, il modello del testo descritto precedentemente. In questo modo è possibile, in un futuro, implementare altri analizzatori di testo ed eventualmente sostituirli senza bisogno di modificare le parti di codice che ne fanno uso. Altri *parser* potrebbero essere implementati per aumentare la precisione degli stessi, per usare altre librerie o per interpretare un'altra lingua.

Altri dettagli sulla costruzione del modello seguono nel capitolo successivo, in particolare verranno spiegati quali algoritmi e strumenti sono stati impiegati per la risoluzione dei riferimenti e l'esplicitazione delle parti sottintese.

3.3 Risoluzione dei riferimenti e parti sottintese

Nel linguaggio naturale è consuetudine riferire a qualcosa di cui si sta parlando in modi differenti. Un esempio notevole è l'uso dei pronomi: invece di ripetere una parte del discorso, questa viene riferita dal pronome. Per esempio:

Es. 3.3.1. "Luca e Tiziana stanno insieme da molto tempo. Ieri, lui le ha chiesto di sposarlo."

In questo esempio ci sono tre pronomi: "lui" che si riferisce a Luca, "le" che intende "a lei", "a Tiziana" e "lo", la parte finale della parola "sposarlo", che sta a significare "sposare lui", "sposare Luca". I pronomi possono sostituire un sostantivo, come nel caso precedente, una aggettivo ("Ti credevo *intelligente*, ma non *lo* sei"), un'intera frase o sintagma ("*Marta mi ha telefonato* e *questo* mi ha fatto molto piacere") o un altro pronome ("Invece del mio profumo ho preso il *tuo*, *che* è più gradevole.").

Possiamo trovare anche altri modi di esprimere riferimenti come gli epiteti, ad esempio: "Andrea è caduto dalla bici. Il genio non aveva controllato i freni!", in questa frase "Il genio", in senso ironico, si riferisce ad Andrea. Oppure: "Gauss dimostrò tantissimi teoremi fondamentali per la matematica, la geometria e la fisica. Il matematico era un tedesco vissu-

to tra l'ottavo e il nono secolo.", "Il matematico" si riferisce ovviamente a Gauss e non ad una generica persona che studia la matematica.

C'è un ulteriore tipo di riferimento più sottile e più complesso che potremmo chiamare "citazione", grammaticalmente non individuabile, ma comprensibile solo a livello semantico. Per esempio: "Alla gara partecipano due squadre: i Potenti e gli Smilzi. La prima con la maglia blu e la seconda con la maglia rossa", "la prima" e "la seconda" si riferiscono rispettivamente alla squadra dei "Potenti" e degli "Smilzi". Oppure "Giacomo, Lucia e Franco sono amici dalle scuole elementari. I tre amici decidono di andare a festeggiare il capodanno a Budapest", "I tre amici" si riferisce a "Giacomo, Lucia e Franco".

In questo lavoro di tesi non è stata affrontata direttamente la risoluzione dei riferimenti degli epiteti o delle citazioni, ma solamente quella dei pronomi e nello specifico il caso in cui sostituiscano un sostantivo. Lo studio dell'uso di tale forma grammaticale ha evidenziato che i pronomi concordano in genere e numero con l'entità che sostituiscono, spesso si riferiscono all'ultimo soggetto espresso e sono declinati a seconda della funzione logica che devono svolgere nel periodo. Riprendendo l'esempio sopra di Luca e Tiziana, entrambi i sostantivi sono i soggetti della prima frase e, giustamente, i pronomi seguenti riferiscono a loro due. Per distinguere i soggetti che riferiscono viene usata la concordanza di genere e numero: "le" è femminile singolare quindi concorda con Tiziana, "lui" e "lo" sono maschili singolari quindi concordano con "Luca". Modificando la prima frase dell'esempio in "Luca sta con Tiziana da molto tempo.", "Tiziana" non è più soggetto della frase, ma, grazie alla concordanza grammaticale, è possibile disambiguare e risolvere comunque i riferimenti dei pronomi successivi: "le" è femminile e non può sostituire il soggetto "Luca" che è maschile quindi si riferisce a "Tiziana", che è femminile, nonostante funga da complemento.

Dallo studio effettuato sulla grammatica dei pronomi è stato sviluppato un algoritmo per la loro risoluzione. Tale algoritmo è schematizzato in modo molto sintetico dal seguente diagramma:

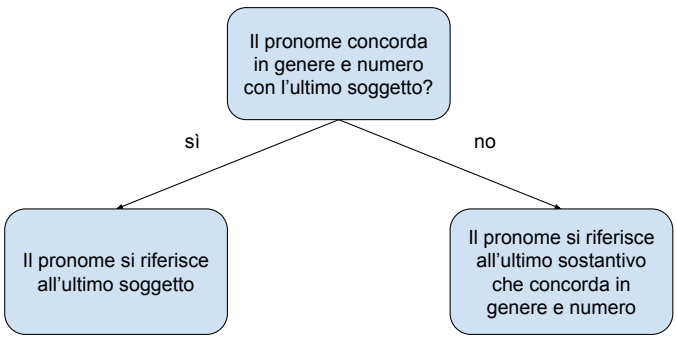


Figura 3.5: Risoluzione dei pronomi

Durante l'analisi del testo tutte le entità che si incontrano vengono salvate in una lista, quando si incontra un pronome si va a cercare nella suddetta lista l'ultimo soggetto, se questo concorda in genere e numero con il pronome allora il soggetto è l'entità che il pronome sostituisce, altrimenti si scorre la lista dall'ultimo al primo elemento fino a trovare quello che corrisponde in genere e numero al pronome. L'entità trovata viene salvata nella proprietà *reference* del pronome.

Per conoscere in dettaglio le proprietà grammaticali dei vocaboli è stato usato il Wikizionario. Tramite la libreria JWKTL, che offre un'interfaccia Java al dizionario, è possibile ottenere le informazioni necessarie per verificare la concordanza. In particolare, data una parola e la sua funzione grammaticale, JWKTL fornisce il genere (maschile, femminile o neutro) e l'elenco delle possibili flessioni della parola, tramite queste ultime è possibile capire il numero.

Trovare la corrispondenza grammaticale tra il pronome e uno dei sostantivi precedenti non è sufficiente, infatti, modifichiamo l'esempio 3.3.1 aggiungendo la frase "Lei accetta e [loro] decidono di andare ai Caraibi per il viaggio di nozze". Il pronome "loro", in questo caso sottinteso, sostituisce "Luca e Tiziana", ma è plurale e tutti i sostantivi precedenti sono singolari. Per ovviare a questo problema viene utilizzata la proprietà *entitiesGroup* di *Entity*. Come spiegato nel capitolo precedente tale proprietà serve a tenere traccia delle entità che vengono elencate tramite congiun-

zione. L'algoritmo sviluppato, quando deve verificare il numero, controlla anche quante entità ci sono nella proprietà *entitiesGroup* e, se ce ne sono più di una, stabilisce che il numero è plurale, così facendo la verifica della corrispondenza tra "loro" e "Luca e Tiziana" ha successo.

L'algoritmo appena descritto è stato sviluppato per affiancare la libreria Stanford CoreNLP. Questa libreria ha un modulo dedicato alla risoluzione dei riferimenti ed è in grado di risolvere pronomi, epiteti e citazioni in generale. Durante la costruzione del modello del testo, ogni qualvolta si incontri una entità, viene consultata questa libreria per ottenere, se esiste, ciò a cui riferisce. Se la libreria Stanford CoreNLP risponde con esisto positivo allora tale riferimento viene aggiunto alla proprietà *reference* dell'entità che si sta analizzando, mentre se non esiste, solo nel caso del pronome, viene invocato l'algoritmo descritto precedentemente.

Lo scopo del modello di testo che si vuole ottenere è quello di rendere esplicito tutto ciò che è possibile per agevolare una successiva analisi mirata ad una specifica interpretazione senza doversi preoccupare di problematiche del linguaggio di basso livello come appunto la risoluzione dei riferimenti e delle parti sottintese. Fino a qui la prima problematica è stata risolta, ora è necessario occuparsi di quelle parti di testo che vengono spesso omesse per evitare ripetizioni o poiché deducibili da una qualche regola grammaticale.

La proposizione è la parte del discorso con senso compiuto più semplice possibile. Per un'analisi del testo sarebbe comodo poter studiare semplicemente tutte le proposizioni ed implementarne una interpretazione. Purtroppo a volte le proposizioni non hanno senso da sole perché dipendono da altre proposizioni. Prendiamo ad esempio la seguente frase "Ci sono cinque monete numerate da 1 a 10.", ricordando la regola che ad ogni verbo corrisponde una proposizione, in questo caso avremo due proposizioni: "Ci sono cinque monete" e "numerate da 1 a 10". Le due proposizioni sono in relazione subordinata, in particolare la seconda dipende dalla prima quindi "numerate da 1 a 10" non ha senso da sola, è necessario aggiungere il soggetto sottinteso "monete" che, in realtà, è nominato nella proposizione principale. Volendo fare tutti i passaggi la frase subisce

le seguenti trasformazioni: ”Ci sono cinque monete *che sono* numerate da 1 a 10.” e, sostituendo il pronome ”che”, diverrebbe ”Ci sono cinque monete, *le cinque monete* sono numerate da 1 a 10.”. Notare che con l'ultimo passaggio le due proposizioni diventano coordinate e quindi indipendenti nel senso che, anche se prese singolarmente, sono di senso compiuto.

È stato sviluppato un algoritmo che attua questa trasformazione e si attiva non appena una proposizione non presenta il soggetto, a questo punto, l'algoritmo cerca l'ultima entità analizzata e la replica come soggetto della proposizione.

Un altro caso di parti del discorso omesse si può notare dal seguente esempio: ”Luigi ha cinque biglie, Cristina sette.”. In questo caso le parti sottintese sono due: il verbo ed il complemento oggetto. La frase può essere riscritta come ”Luigi ha cinque biglie, Cristina ne ha sette.” e ancora ”Luigi ha cinque biglie, Cristina ha sette biglie.”. Per questo tipo di omissioni l'algoritmo si attiva quando viene espressa una quantità senza che ci sia la sua unità di misura. In tal caso si cerca semplicemente la penultima entità analizzata e la si replica con lo stesso ruolo logico dell'entità trovata.

3.4 Conclusioni

In questo capitolo è stato affrontato il tema dell'analisi del linguaggio naturale, si sono visti gli strumenti che sono stati utilizzati e gli algoritmi implementati per risolvere alcune problematiche. Nonostante la potenza e la complessità delle librerie usate è stato comunque necessario implementare alcune procedure in modo autonomo. Si pensi che Stanford CoreNLP non sempre genere un albero sintattico corretto ed anche la risoluzione dei riferimenti a volte fallisce. JWKTL, la libreria che si interfaccia al Wikizionario, presenta anche lei qualche imprecisione. Il risultato è buono, ma limitato e a volte fallace. Considerando solo i pronomi, se si dovessero risolvere nel caso di sostituzione di aggettivi o, peggio ancora, nel caso di sostituzione di un'intera frase, ce ne sarebbe ancora molto di lavoro da fare. Ancora più complesso il discorso sugli epiteti e sulle citazioni. In ogni caso, per lo scopo di questo lavoro di tesi, il modello prodotto da questa

fase è adeguato alla complessità di una buona parte dei quesiti matematici di interesse.

La fase successiva prevede l'analisi semantica del modello di testo al fine di produrre un modello matematico. Prima di affrontare questo tema, nel prossimo capitolo, si discuterà di alcuni principi realizzativi utilizzati allo scopo di rendere tale analisi più agevole.

Capitolo 4

Specifiche implementative e pattern utilizzati

Prima di affrontare i capitoli riguardanti l'elaborazione del testo finalizzata all'analisi semantica e quindi alla deduzione del modello matematico è bene descrivere quali strumenti sono stati utilizzati per rendere il codice modulare, flessibile, manutenibile ed estendibile. I concetti utilizzati sono propri dell'ingegneria del software ed indispensabili per un'applicazione di grandi dimensioni e con molte funzionalità. L'utilizzo di queste tecniche ha permesso e permetterà in futuro di espandere e modificare l'applicazione senza la necessità di stravolgere la struttura generale.

4.1 Interfacce

Il primo concetto che è stato utilizzato è riconducibile all'uso delle interfacce. Le interfacce consentono di nascondere l'implementazione dei componenti. In questo modo è possibile avere diverse implementazioni di una stessa interfaccia e sostituire tali implementazioni senza influenzare il resto del sistema software.

Tale concetto è stato applicato in questo lavoro di tesi per separare i macro moduli di analisi generica del testo da quelli di analisi semantico-matematica e i componenti minori all'interno dei suddetti macro moduli. Un caso di applicazione di questo concetto è stato citato nel capitolo

precedente descrivendo le classi *Parser*, ma è possibile trovarlo anche per quanto riguarda i dizionari ed i generatori dei modelli matematici.

4.2 Il pattern visitor

Il modello del testo ed il modello matematico Java, che verrà descritto nei capitoli successivi, sono strutture piuttosto complesse che spesso presentano gerarchie di classi articolate. Per poter navigare in tali strutture è doveroso evitare procedure fragili, brutali o poco leggibili come liste smisurate di costrutti *if-else* o *cast* forzati. Si pensi alla classe *Quanity*, descritta nel capitolo precedente, che presenta ben sei classi derivate. Quando è necessario analizzare un'istanza che sembra di una certa classe bisogna sapere che sottoclasse è in realtà per poterla interpretare in modo corretto. Per risolvere tale problematica è stato utilizzato il *pattern visitor*.

Il *pattern visitor* viene implementato come segue: nelle classi che è necessario esaminare e di cui bisogna conoscere l'istanza viene implementato un metodo di accettazione che ha come argomento un visitatore. Il visitatore, rappresentato da un'interfaccia, sarà il componente che vuole ispezionare l'oggetto. L'interfaccia del visitatore deve avere un metodo per ogni sottoclasse. Quando il visitatore vuole ispezionare un oggetto, chiama il metodo di accettazione e, con il meccanismo di inversione di controllo e sfruttando l'*overloading*[1] di Java, viene invocato sul visitatore il metodo corrispondente alla classe propria dell'oggetto.

Il vantaggio di questa tecnica consiste, ancora una volta, nel separare un componente che ne usa un altro. Permette di creare più oggetti, ognuno con una funzione diversa, applicabili in modo indipendente ad una stessa struttura dati. In questo lavoro di tesi i componenti che costruiscono il modello matematico sono i visitatori del modello di testo, ma sono stati implementati altri visitatore che, per esempio, stampano delle informazioni sul modello di testo a scopo di *debug*. Si pensi che, con tale architettura,

[1] L'overloading è quel meccanismo per il quale anche se una classe ha più metodi con la stessa firma, ma argomenti di tipo o numero diverso, viene invocato il metodo corretto a seconda degli argomenti passati nella chiamata.

è possibile in futuro realizzare un altro modulo in grado di costruire un altro tipo di modello matematico dall'analisi del modello del testo senza influenzare le altre parti del programma.

4.3 Algoritmi per l'analisi semantica

Allo scopo di effettuare un'analisi semantica sarebbe utile avere una sorta di database dal quale, data una parola, si possa ricavare il significato di quest'ultima. Riflettendoci, questo ideale database, data una parola, potrebbe restituire come significato la parola stessa, poiché è vero che questa rappresenta il suo stesso significato. In realtà la semantica delle parole, o la semantica in generale di una qualsiasi rappresentazione (sonora, visiva, gestuale etc.), è l'azione che ne scaturisce da un'entità che è in grado di interpretarla. Ad esempio se la madre di Pierino chiedesse a suo figlio di andarle a prendere il latte, si può verificare che Pierino ha compreso il significato di quelle parole se, entro un certo periodo, egli facesse avere a sua madre il latte. Nel caso di questo lavoro di tesi, il sistema software capisce il testo di un problema matematico se è in grado di generarne un modello matematico che lo risolva.

L'idea che sta alla base della costruzione del modello matematico, in questa tesi, si basa sulla comprensione, e quindi sull'interpretazione, di un numero limitato di concetti. Poiché si lavora sul testo scritto, i concetti sono espressi da vocaboli, ma vocaboli diversi possono riferire a concetti comuni, ad esempio "avere" e "possedere" sono due verbi che esprimono lo stesso concetto: il concetto di "appartenenza". Ciò che serve, quindi, è uno strumento intermedio che riesca a ridurre le molteplici rappresentazioni di un concetto ad un'unica rappresentazione conosciuta e interpretabile dai generatori del modello matematico.

Lo strumento semantico che vogliamo ottenere deve funzionare nel seguente modo: dato l'insieme dei concetti conosciuti, rappresentato dall'insieme dei vocaboli C e dato l'insieme di tutte le parole P appartenenti al dizionario di una lingua specifica, il componente semantico deve essere in grado di associare ad una parola $p \in P$ il corrispondente vocabolo $c \in C$

tale che il concetto associato a *p* sia simile al concetto associato a *c* secondo una certa funzione di similarità.

È normale e conveniente che la cardinalità di *C*, ovvero dell'insieme dei concetti interpretabili, sia di molto inferiore alla cardinalità di *P*. Si noti che l'insieme *C* deve essere sempre uguale a prescindere dalla lingua a cui appartiene l'insieme *P*, in questo modo gli utilizzatori di questo strumento semantico possono rimanere indipendenti dalla lingua.

Illustriamo un piccolo esempio sulla questione dell'indipendenza linguistica. Se nell'insieme dei concetti interpretabili esiste il concetto di "appartenenza", rappresentato proprio dalla parola "appartenenza", è possibile dare in ingresso allo strumento semantico il verbo "avere" o il verbo "possedere" ed ottenere lo stesso risultato, ovvero la parola "appartenenza", ma lo stesso risultato deve essere restituito anche se in ingresso viene dato il termine in inglese "have", poiché anche quest'ultimo rappresenta il concetto di "appartenenza". Per motivi di semplificazione realizzativa è comunque meglio creare diverse implementazioni dello strumento semantico, una per ogni lingua.

A livello implementativo lo strumento semantico suddetto si riduce ad una classe, o meglio ad un'interfaccia *ISemantic*, con il suo metodo *String meaning(String word)* che associa ad una parola (*word*) il vocabolo (valore di ritorno del metodo) che rappresenta un noto concetto. La funzione di similarità, invece, è implementata applicando i concetti di iperonimia.

Entrando ancora più nel dettaglio, i concetti conosciuti sono espressi da una lista di metodi nella forma *float meansSomething(String word)* dove *Something* è il vocabolo che rappresenta il concetto conosciuto. Il metodo restituisce il valore di similarità tra il significato associato a *Something* ed il significato associato alla parola *word*. In più sono presenti dei metodi generici per calcolare la similarità tra due vocaboli. L'interfaccia *ISemantic* si presenta come segue:

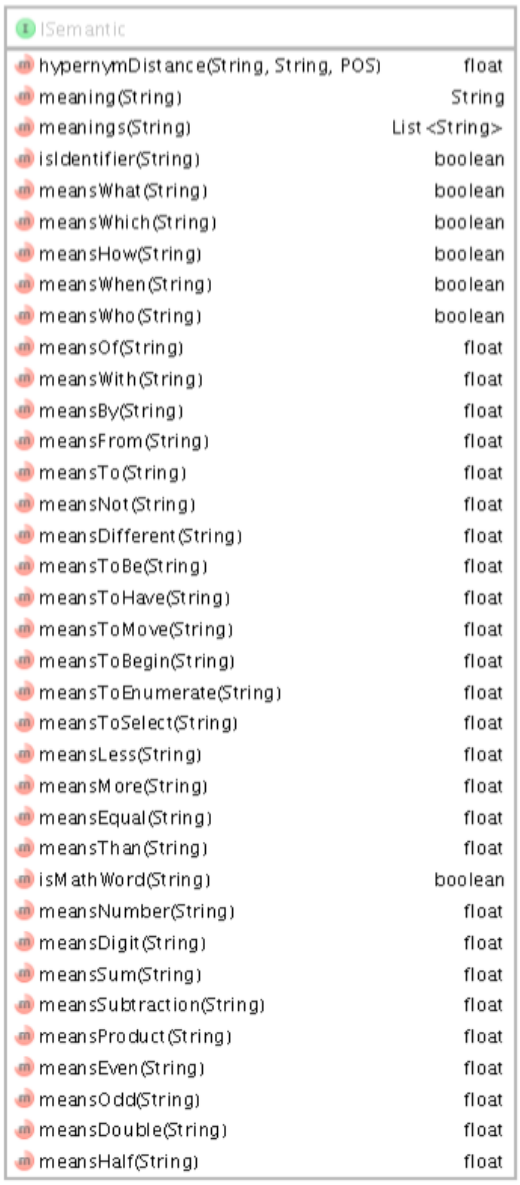


Figura 4.1: L'interfaccia ISemantic

La classe realizzata che implementa questa interfaccia è chiamata *WordNetSemantic* e fa uso della libreria WordNet. Il metodo *float hypernymDistance(String word, String hypernym)* corrisponde alla funzione di dissimilarità basata sull'iperonimia, in particolare *hypernym* esprime un concetto più generico di *word* tanto maggiore è il valore di ritorno.

Il calcolo di questa funzione si basa sulla navigazione del grafo di *syn-*

set di WordNet. Tale funzione restituisce il valore di dissimilarità tra due parole, questo valore si basa sulla distanza dei due concetti messi in relazione di iperonimia. Data una parola, la prima, la libreria restituisce i *synset* in ordine statistico di attinenza alla parola stessa, quindi, più ci si allontana dal primo *synset* più la dissimilarità aumenta. Mano a mano che si sale il grafo, verso concetti più generici (iperonimia), la dissimilarità aumenta ancora. Non appena si incontra il *synset* della seconda parola ci si ferma e si restituisce il valore di dissimilarità calcolato.

Gli altri metodi dell'interfaccia *ISemantic* nella forma *float meansSomething(String word)* fanno uso della funzione di dissimilarità cablando uno dei due argomenti, ad esempio *float meansToMove(String word)* è implementato chiamando *return hypernymDistance(word, "transfer", POS.VERB);*.

Ci sono ora tutti gli elementi per descrivere l'implementazione del metodo principale, *String meaning(String word)*. La sua realizzazione, tramite *reflection*[2], invoca tutti i metodi di *ISemantic* che iniziano con "means" passando come argomento *word*. Il metodo che restituisce il valore di dissimilarità più basso è il metodo vincitore. Il valore di ritorno di *String meaning(String word)* è il nome del metodo vincitore meno il prefisso "means", in questo modo è possibile restituire la parola corrispondente al concetto. Ad esempio, se il metodo vincitore fosse *float meansToMove(String word)* il valore di ritorno sarebbe "ToMove". Il vantaggio di usare la *reflection* consiste nel fatto che per aggiungere un nuovo significato è sufficiente aggiungere il rispettivo metodo.

Il componente semantico così realizzato consente di essere utilizzato in modo molto semplice e rapido. Tipicamente è possibile mappare i concetti noti ad "azioni" tramite delle semplici tabelle. L'architettura che è stata pensata per usare in modo efficace questo strumento semantico è la seguente:

[2]La reflection è quella funzionalità per la quale, a *runtime*, è possibile ispezionare le proprietà ed i metodi delle classi.

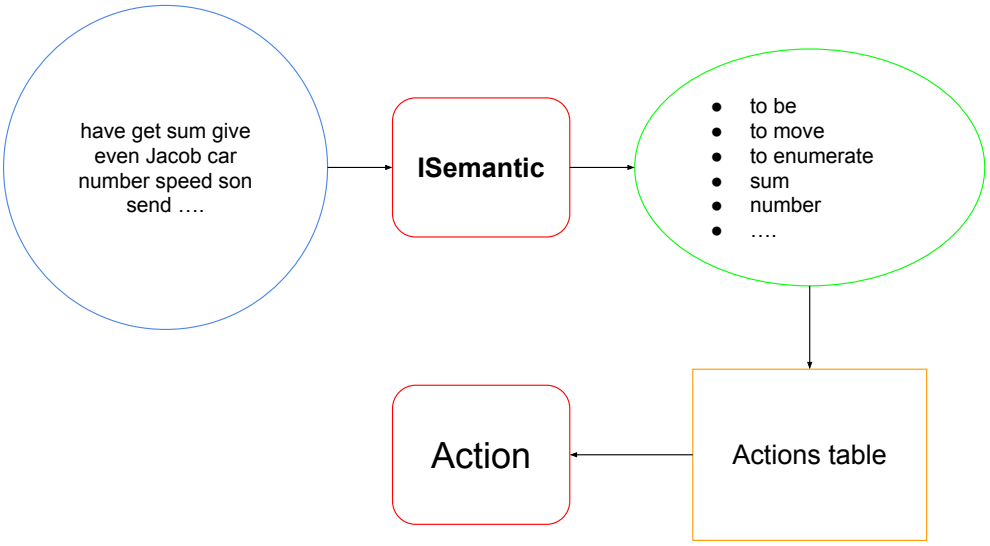


Figura 4.2: Architettura per l'interpretazione semantica

Tale struttura è implementata nei moduli che si occupano di creare il modello matematico. Da qui in poi sarà implicito che i generatori dei modelli ricavino i concetti che sono in grado di interpretare tramite l'architettura di cui sopra. Ad esempio se nel testo si incontrassero i verbi "muovere", "spedire", "trasferire", "inviare", "dare" etc. questi sarebbero ricondotti al concetto di "muovere qualcosa da un'entità ad un'altra" e tale concetto associato, tramite tabella, ad un'azione specifica spesso implementata da un metodo particolare.

4.4 Conclusioni

L'uso di interfacce, pattern dell'ingegneria del software e *reflection* ha permesso di avere un'implementazione pulita, modulare, estendibile e leggibile. Si pensi se si fosse usato il costrutto *if-else* per discriminare i significati delle parole: il codice sarebbe risultato molto complicato, lungo e laborioso da modificare ed estendere.

Nei prossimi capitoli vedremo come vengono generati i modelli matematici concentrandosi sulla logica che guida l'interpretazione e tralascian-

do gli aspetti implementativi già descritti in questo capitolo. In particolare verranno presentati due componenti per la modellazione matematica: uno per la deduzione di modelli logico-matematici dei quali si ricava una rappresentazione descritta in Prolog, l'altro per la risoluzione di problemi aritmetici dai quali viene ricavato un sistemi di equazioni lineari.

Capitolo 5

Deduzione del modello logico-matematico

Dall'archivio dei problemi matematici della Bocconi[12] si può notare che la gran parte di questi quesiti sono di tipo logico-matematici, di conseguenza la programmazione logica è un ottimo modo per modellare questo tipo di problemi. Il principio alla base della trasformazione del testo nel modello logico è ispirato al lavoro di Patrick Blackburn e Johan Bos[7] sulla comprensione semantica del testo in linguaggio naturale e consiste nell'analizzare i verbi di ogni proposizione e per ognuno applicare una interpretazione logico-matematica che metta in relazione le entità coinvolte. Quello che vogliamo andare a costruire è un modello logico scritto in Prolog. Per fare ciò devono essere identificate le variabili ed i predicati. L'idea di base è quella di analizzare il testo e, mano a mano, comporre una lista di predicati che andranno a costruire il goal che verrà poi risolto da un interprete Prolog. Il risultato dell'esecuzione restituirà in uscita il valore di tutte le variabili logiche create e, tramite l'analisi della domanda del problema, si potrà poi individuare quali sono le variabili che contengono la soluzione richiesta.

I quesiti matematici disponibili sono stati studiati e risolti deducendo manualmente il modello logico. Da questo lavoro è emerso che molti termini Prolog sono ricorrenti e possono essere riutilizzati per risolvere più problemi, in più sono state rilevate delle categorie di verbi e delle entità

significative che si ripetono nei diversi quesiti. Questo studio ha permesso di individuare un insieme di verbi ed entità sulle quali implementare una specifica semantica logico-matematica. In particolare le entità sono *sum*, *difference*, *product*, *number*, *digit*, *even*, *odd*, *double*, *half*, *twice*, mentre i verbi sono *to be*, *to have*, *to start*, *to number*, *to select*, *to multiply*, *to divide*, *to sum*, *to subtract*.

Dove l'analisi specifica non è implementata il modello viene costruito applicando un algoritmo generale che consistente nel trasformare le entità, ovvero i soggetti e i complementi, in variabili logiche con i nomi uguali o comunque contenenti il nome delle entità da cui derivano ed i verbi in predicati che mettono in relazione le variabili ricavate dalle entità.

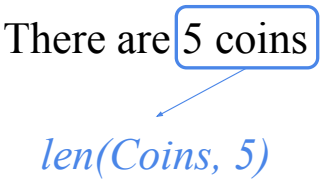
Riportiamo di seguito un esempio di quesito logico-matematico che accompagnerà la descrizione dei passaggi utili alla costruzione del modello:

Es. 5.0.1. ″There are 5 coins numbered from 1 to 5. Jacob chooses two coins so that their sum is an even number. In how many ways can he choose them?″

Ricordando il capitolo 3 il modello di testo è composto da frasi a loro volta costituite da proposizioni, queste ultime completate e modificate di modo da avere tutte le parti esplicitate: le parti sottintese riscritte esplicitamente e i pronomi sostituiti dalle entità a cui si riferiscono. Dall'esempio di riferimento vengono generate le seguenti proposizioni:

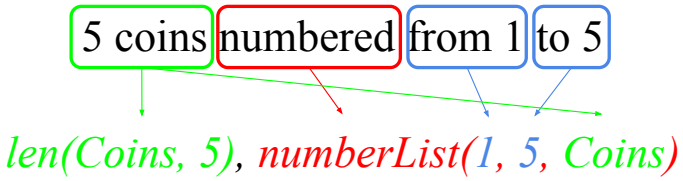
1. ″There are 5 coins″
2. ″5 coins numberd from 1 to 5″
3. ″Jacob chooses two coins″
4. ″so that two coins sum is an even number″
5. ″In how many ways can he choose coins?″

La prima proposizione è molto semplice e genere un solo predicato ricavato dall'entità ″5 monete″:



Questa produzione si attiva ogniqualvolta una entità ha un quantificatore, in particolare se la semantica indica una quantità esatta il predicato corrispondente è *len/2*, se la quantità espressa è "maggiore di *x*", "maggiore o uguale a *x*", "minore di *x*" o "minore o uguale a *x*", ponendo *Entity* uguale al nome dell'entità in esame, *X* uguale al valore di *x*, *NextX* uguale a *x+1* e *PreviousX* uguale a *x-1*, i relativi predicati sono *minLen(Entity, NextX)*, *minLen(Entity, X)*, *maxLen(Entity, PreviousX)*, *maxLen(Entity, X)*. I predicati *len/2*, *minLen/2* e *maxLen/2* e i predicati che verranno usati nel seguito sono stati dedotti dallo studio dei problemi matematici di cui sopra e definiti in un file Prolog che deve essere caricato dal risolutore prima di procedere alla risoluzione del problema. Questi tre predicati appena citati sono la mappatura in linguaggio logico delle istanze di classe *EqualQuantity*, *GreaterQuantity*, *GreaterOrEqualQuantity*, *SmallerQuantity* e *SmallerOrEqualQuantity* spiegate nel capitolo 3.1. Il nome della variabile deriva direttamente dal nome dell'entità modificata in modo che abbia la prima lettera maiuscola, come la sintassi Prolog richiede per il nome delle variabili, e la "s" finale. Applicare una certa regola nella generazione dei nomi delle variabili aiuta a costruire un modello coerente: alle entità che hanno lo stesso nome deve corrispondere una stessa variabile.

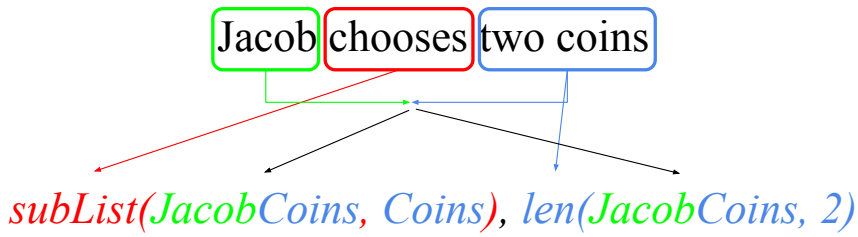
La seconda proposizione è tradotta come segue:



Il soggetto, in verde, è esattamente la stessa entità "5 coins" della proposizione precedente, riscritta poiché sottintesa. Processata dall'algoritmo, genera un duplicato del predicato *len/2* prodotto dalla proposizione precedente, ma la cosa non è un problema: anche se ridondante non comprometterà l'esito della risoluzione e, se necessario, potrà essere eliminato in una fase successiva.

Il verbo "numbered" rientra in una categoria di verbi a cui è stata specificata una semantica, in particolare, a tale categoria, corrisponde il predicato *numberList(Min,Max,Numbers)*. Questo predicato genera una lista (*Numbers*) di numeri interi che vanno da *Min* a *Max*. Di fatto, questo predicato, può essere visto come una sorta di dominio, infatti sui valori di tale lista verranno fatti tutti i tentativi per risolvere il quesito. Il verbo *to number*, e tutti quelli con pari significato, si aspetta un complemento che indichi l'inizio della numerazione e/o uno che ne indichi la fine. Le quantità di questi complementi, se presenti, andranno a sostituire rispettivamente le variabili *Min* e *Max* del predicato *numberList/3*, altrimenti verranno create delle variabili di nome *EntityMin* e *EntityMax*. Il nome della lista viene ricavato invece dal soggetto, in questo caso *Coins*, generato secondo la regola spiegata sopra.

La proposizione successiva, la terza, contiene il verbo *to chooses*, anche questo appartenente ad una categoria nota. L'interpretazione che ne è stata associata si concretizza nel predicato *subList/2*, ovvero scegliere qualcosa corrisponde, in termini di modello logico, selezionare un sottoinsieme da un altro insieme, in Prolog: selezionare una sottolista da una lista.

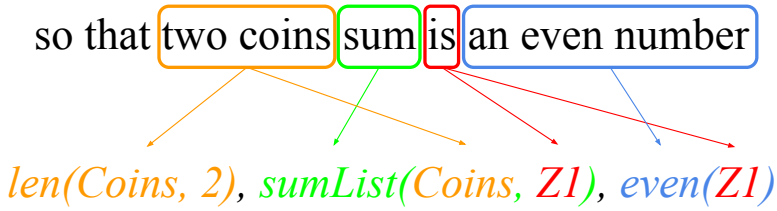


Tralasciando il già discusso predicato *len/2* generato, come nei casi pre-

cedenti, dall'entità "two coins", concentriamoci sulla variabile *JacobCoins*. Come viene costruita tale variabile? L'algoritmo è comune anche ad altri verbi ed il principio è quello di unire il nome del soggetto della proposizione con il nome del complemento in modo da avere una variabile univoca. In questo caso all'entità "two coins" è associata la variabile *JacobCoins*, si tenga a mente questa relazione poiché sarà indispensabile più avanti.

Riassumendo, l'analisi di questa proposizione genera due predicati Prolog che producono una sottolista di *Coins* di lunghezza due.

L'ultima proposizione, senza contare la domanda finale, è tradotta come segue:



Ricordando che "two coins" è la sostituzione del pronome "their", anche in questo caso abbiamo la produzione ridondante del predicato *len(Coins, 2)* che, come già spiegato, non crea problemi e può essere successivamente eliminato. In questa proposizioni abbiamo diverse cose interessanti a partire dai sostantivi "sum", "number" e l'aggettivo "even", che rientrano nelle entità con semantica specificata, al verbo essere, anch'esso con la sua particolare interpretazione.

Per quanto riguarda l'analisi delle entità c'è da dire che, in generale, la loro traduzione può essere di due tipi: trasformazione in semplice variabile logica o trasformazione in predicato logico. In particolare la generazione di un predicato avviene solo quando l'entità è nota e quindi ne è associata una semantica. Nelle proposizioni precedenti i sostantivi "coins" e "Jacob" non avevano una particolare interpretazione quindi sono stati tradotti in semplici variabili, in questo caso, invece, "sum", "number" e "even" hanno una particolare semantica matematica e quindi vengono tradotti nei rispettivi predicati.

Il sostantivo ”sum” è associato al predicato *sumList/2* che, come dice il nome stesso, esegue la somma degli elementi della lista e la assegna al secondo argomento. L'aggettivo ”even”, affiancato al sostantivo ”number”, genera il predicato omonimo *even/1* che è verificato se l'argomento è un numero pari.

Di particolare interesse è il verbo essere. Solitamente questo verbo implica il concetto di uguaglianza tra il soggetto ed il complemento oggetto oppure qualifica il soggetto tramite un aggettivo. Ad esempio: ”La somma è un numero pari” o ”Il numero è pari”. L'uguaglianza in Prolog può essere espressa proprio dal predicato predefinito *is* oppure, più semplicemente, usando la stessa variabile logica. Si noti infatti che nel modello sopra la somma delle monete sarà contenuta nella variabile *Z1* ed il numero pari, espresso dal predicato *even/1*, è altrettanto contenuto nella variabile *Z1*. Ciò implica che la variabile *Z1* deve essere la somma di *Coins* e deve essere anche un numero pari. Il verbo ”essere”, quindi, viene tradotto mediante l'uso della stessa variabile logica tra il soggetto ed il complemento oggetto o tra il soggetto e l'aggettivo che segue il predicato.

Il verbo essere ha anche altre interpretazioni, per esempio ”La somma dei pettorali è minore del prodotto dei pettorali”, in questo caso non viene espressa un'identità, bensì una disuguaglianza, quindi l'interpretazione è completamente differente: le variabili per la somma e per il prodotto sono differenti ed il vincolo è tradotto semplicemente con l'operatore corrispondente ”$<$”. La traduzione completa sarebbe:

$$sumList(Pectorals, PectoralsSum),$$

$$productList(Pectorals, PectoralsProduct),$$

$$PectoralsSum < PectoralsProduct$$

Riprendendo l'esempio di riferimento e mettendo insieme i pezzi si

ottiene il seguente modello logico-matematico:

$$len(Coins, 5),$$

$$len(Coins, 5), numberList(1, 5, Coins),$$

$$subList(JacobCoins, Coins), len(JacobCoins, 2),$$

$$len(Coins, 2), sumList(Coins, Z1), even(Z1).$$

Il modello generato è sbagliato e fallisce l'esecuzione. L'errore risiede nell'interpretazione troppo semplice dell'ultima proposizione: si sta parlando di monete, ma bisogna distinguere tra le monete scelte da Jacob e le generiche monete presenti. La traduzione che è stata fatta fino ad adesso opera ciecamente sulle parole del testo e le trasforma brutalmente in variabili e predicati. Le monete nominate all'inizio del testo non sono le stesse nominate alla fine: le prime sono le cinque monete che Jacob può scegliere e le seconde sono le monete che Jacob sceglie. Lo stesso sostantivo indica due entità ben distinte e questa distinzione deve riflettersi anche sul modello. Per fare ciò è stato implementato un sistema che tiene traccia dell'associazione tra entità e variabile logica associata.

Per spiegare il meccanismo è necessario ricordare come è composta la proposizione e più precisamente come avviene il completamento delle parti mancanti e la sostituzione dei riferimenti. Quando c'è un soggetto sottinteso o un pronome, viene creato un oggetto di classe *Entity* che come proprietà *reference* ha l'entità a cui riferisce. In questo esempio, l'entità "their" dell'ultima proposizione è un oggetto di classe *Entity* con la proprietà *reference* che punta all'entità "two coins".

Durante l'analisi delle proposizioni, le variabili logiche che vengono create, derivanti dalle entità, sono salvate in una struttura dati che memorizza tale associazione. Se un'entità ha l'attributo *reference* diverso da *null* non viene creata una nuova variabile, ma viene utilizzata quella associata alla entità in *reference*. Nel caso in esame, nella terza proposizione, a "two coins" viene associata la variabile *JacobCoins*. Nella quarta proposizione,

l'entità "their" ha la proprietà *reference* uguale a "two coins" alla quale a sua volta è associata la variabile *JacobCoins*, quindi viene usata quest'ultima variabile anche per l'entità "their". Lo schema seguente illustra i riferimenti esistenti tra le entità e le variabili, le frecce blu rappresentano la proprietà *reference* e quelle verdi l'associazione con la variabile:

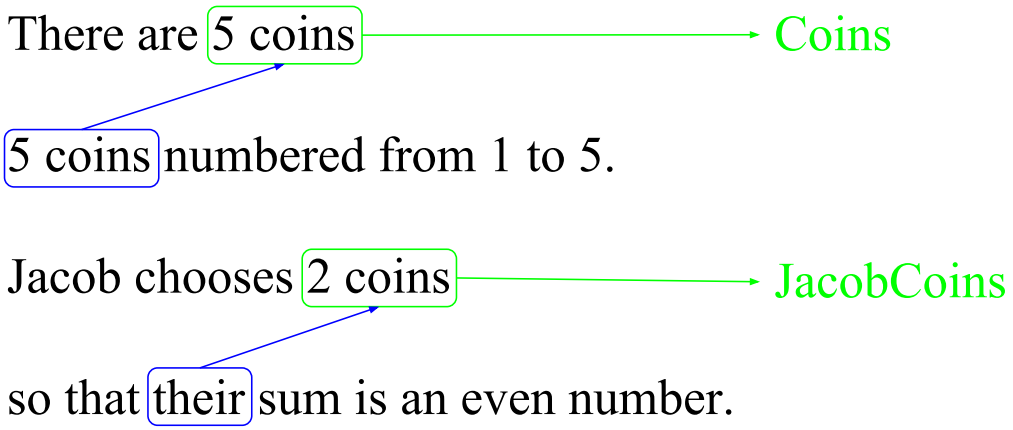


Figura 5.1: Associazione tra entità e variabili logiche

Poiché, quando possibile, viene riutilizzata una variabile creata precedentemente, non è necessario processare nuovamente l'entità a cui si riferisce, di conseguenza anche il problema della ridondanza dei predicati *len/2* correlati ai quantificatori delle entità è risolto.

A questo punto il modello risultante è coerente e logicamente corretto:

$$len(Coins, 5),$$

$$numberList(1, 5, Coins),$$

$$subList(JacobCoins, Coins), len(JacobCoins, 2),$$

$$sumList(JacobCoins, Z1), even(Z1).$$

Prima di farlo risolvere da un interprete Prolog è necessario apportare un ultimo accorgimento tecnico. Il predicato *len/2* deve avere il primo argomento di tipo lista già inizializzato altrimenti fallisce, per questo motivo

è stato implementato un algoritmo che riordina i predicati ed in particolare posiziona *len/2*, *maxLen/2* e *minLen/2* in coda.

Il programma Prolog finale è il seguente:

$$numberList(1, 5, Coins),$$

$$subList(JacobCoins, Coins),$$

$$sumList(JacobCoins, Z1), even(Z1),$$

$$len(Coins, 5),$$

$$len(JacobCoins, 2).$$

Per l'esecuzione del programma viene usato l'interprete SWI-Prolog[18]. Prima dell'esecuzione è necessario caricare un modello con i predicati predefiniti come *len/2*, *numberList/3* etc.. Il risultato dell'elaborazione ritorna con successo tutte le possibili soluzioni del problema:

```
Coins = [1, 2, 3, 4, 5],
JacobCoins = [1, 3],
Z1 = 4 ;
Coins = [1, 2, 3, 4, 5],
JacobCoins = [1, 5],
Z1 = 6 ;
Coins = [1, 2, 3, 4, 5],
JacobCoins = [2, 4],
Z1 = 6 ;
Coins = [1, 2, 3, 4, 5],
JacobCoins = [3, 5],
Z1 = 8 ;
false.
```

5.1 Dettagli implementativi

Come spiegato nel capitolo 4 la struttura di questo componente si basa su quella illustrata in figura 4.2. In particolare la tabella delle azioni è divisa in due tabelle, una per una prima analisi dei verbi e l'altra per una successiva trasformazione di verbi ed entità in predicati Prolog. La prima tabella è implementata usando la *reflection*, ovvero dato il significato del verbo, ottenuto da una istanza di *ISemantic*, viene invocato il metodo specializzato nella traduzione di quella categoria di verbi. La seconda tabella, invece, è stata realizzata con una semplice *Map¡String,String¿* della libreria standard di Java. Le chiavi di questa tabella sono i concetti ricavabili da *ISemantic* e i valori sono i nomi dei predicati Prolog con cui mappare i concetti stessi. Di conseguenza, per estendere il componente, è sufficiente implementare nuovi metodi per nuove categorie di verbi o aggiungere mappature tra concetti e predicati Prolog. La traduzione in predicati è più complessa per quanto riguarda i verbi poiché la cardinalità di questi predicati può variare di molto, mentre è più semplice per quanto riguarda le entità per le quali la traduzione corrisponde molto spesso a predicati di cardinalità uno o due e sono facilmente deducibili tramite un unico algoritmo generico.

Per una maggiore semplicità e coerenza sintattica quasi tutti i predicati prodotti sono nella forma prefissa, anche quelli predefiniti come +, -, * e / che prevedono anche la sintassi infissa. Alcuni sono stati anche rimappati in predicati ad-hoc per poter essere più flessibili e quindi usati indifferentemente per variabili di tipo lista o variabili semplici.

Di seguito alcuni dei principali predicati creati per modellare le parti che compaiono più frequentemente nei quesiti matematici. *len/2* e le sue varianti *maxLen/2* e *minLen/2* allo scopo di mappare i quantificatori delle

entità:

$$len([], 0) :- !.$$

$$len([\_], 1) :- !.$$

$$len([\_|T], L) :- len(T, TL), L\ is\ TL + 1.$$

$$minLen([H|T], Min) :- len([H|T], Len), Len >= Min.$$

$$maxLen([H|T], Max) :- len([H|T], Len), Len =< Max.$$

La somma di due numeri implementata di modo che qualsiasi sia l'incognita, uno dei due addendi o il risultato, riesca comunque a dare una soluzione:

$$sum(X, Y, Z) :- integer(X), integer(Y), !, Z\ is\ X + Y.$$

$$sum(X, Y, Z) :- integer(Z), integer(Y), !, X\ is\ Z - Y.$$

$$sum(X, Y, Z) :- integer(Z), integer(X), !, Y\ is\ Z - X.$$

Il concetto di sommatoria, facendo attenzione alla associatività, e produttoria che corrispondono alla traduzione dei sostantivi *sum* e *product* o dei

verbi *to add* e *to multiply*:

$sumList([], 0) :- !.$

$sumList([H], H) :- !.$

$sumList([H1, H2], Sum) :- Sum\ is\ H1 + H2, !.$

$sumList([H1, H2|T], Sum) :- HSum\ is\ H1 + H2, sumList([HSum|T], Sum).$

$productList([], 0) :- !.$

$productList([H], H) :- !.$

$productList([H|T], Z) :- productList(T, Y), Z\ is\ H * Y.$

Il predicato *numDigits/2* che converte un numero da formato intero a formato stringa e fa uso dei predicati predefiniti `atom_number/2` e `number_chars/2` di SWI-Prolog:

$charsNumbers([Char], [Digit]) :-$

$\texttt{atom\_number}(Char, Digit).$

$charsNumbers([HC|TC], [HD|TD]) :-$

$\texttt{atom\_number}(HC, HD), charsNumbers(TC, TD).$

$numDigits(Num, Digits) :- charsNumbers(Chars, Digits), \texttt{number\_chars}(Num, Chars).$

L'interpretazione del concetto di "selezione" corrispondente al predicato *subList/2* che può essere usato assieme ai predicati *len/2* e varianti per ottenere tutte le possibili sottoliste di lunghezza esattamente uguale, maggiore

o minore a quanto richiesto dal testo del problema:

$$subList([], \_).$$

$$subList([X|XS], [X|XSS]) :- subList(XS, XSS).$$

$$subList([X|XS], [\_|XSS]) :- subList([X|XS], XSS).$$

Una particolare considerazione è da dedicarsi al predicato *numberList/3* definito di seguito:

$$numberList(Num, Num, [Num]) :- !.$$

$$numberList(Min, Max, [Min|T]) :-$$

$$Min < Max, Next\ is\ Min + 1, numberList(Next, Max, T).$$

$$numberList(Max, Min, [Max|T]) :-$$

$$Min < Max, Next\ is\ Max - 1, numberList(Next, Min, T).$$

Tecnicamente tale predicato genera semplicemente una lista ordinata, in senso sia crescente che decrescente, di numeri interi da *Min* a *Max* compresi. Concettualmente, però, può essere visto come la definizione di un dominio ed è indispensabile usarlo nella quasi totalità dei problemi in esame.

5.2 Conclusioni

Il modello logico in Prolog si adatta molto bene a questo tipo di problemi matematici, l'interpretazione del testo è abbastanza lineare e di facile implementazione. Una riflessione particolare va fatta sul predicato *numberList/3*. Come spiegato precedentemente, questo predicato rappresenta concettualmente la definizione di un dominio ed è utile nonché necessario usarlo in tutti i problemi. Di conseguenza sarebbe conveniente estende-

re il modello ad una rappresentazione a vincoli per avere una maggiore potenza espressiva e maggiori strumenti di modellazione. Si prenda ad esempio l'esercizio risolto in questo capitolo: la parte del problema riguardante la definizione delle variabili può essere espressa in modo più chiaro ed efficiente nel modo seguente:

$$JacobCoins = [X1, X2],$$

$$JacobCoins\ ins\ 1..5,$$

$$all\_different(JacobCoins)$$

Per quanto riguarda l'interpretazione semantica si è visto che anche in quesiti in cui non si parli di concetti solo matematici è possibile ricavare un modello più o meno corretto. Si prenda ad esempio l'esercizio trattato in questo capitolo: il modulo che deduce il modello non sa minimamente chi o cosa sia Jacob o cosa siano le monete, ma riesce, solo conoscendo i termini che riferiscono alla matematica (ad esempio somma, numero pari, etc.), a costruire un modello che risolve il problema. La capacità di comprensione di questo componente si limita alle relazioni semantiche di iperonimia che permettono di capire e collegare concetti più o meno generici l'uno rispetto ad un altro così da poter, ad esempio, generare variabili coerenti nel caso in cui due sostantivi differenti vogliano riferire alla stessa cosa. Concetti semanticamente più difficili da collegare come ad esempio il fatto che i partecipanti ad una gara abbiano un pettorale e che quest'ultimo sia un numero univoco distribuito in ordine di iscrizione non vengono risolti. Cablare questo tipo di conoscenza potrebbe essere possibile, ma richiederebbe uno sforzo implementativo enorme tanto da far pensare ad una via alternativa come ad esempio algoritmi di apprendimento automatico. Ne rimandiamo la discussione nelle conclusioni di questa tesi poiché tale problematica è comune a più parti di questo lavoro di tesi.

Capitolo 6

Deduzione del modello matematico

Per alcune tipologie di problemi, o parte di essi, è necessario ricavare un modello matematico di tipo aritmetico, ovvero basato su sistemi di equazioni, solitamente lineari, risolvibile per via analitica. Con un modello di questo tipo è potenzialmente possibile risolvere una vasta gamma di quesiti matematici e fisici, da semplici situazioni di calcolo di numeri di entità a descrizione di realtà fisiche quali lo spostamento, la velocità e accelerazione di un'automobile, ad esempio.

La deduzione di questo modello matematico è stata affidata ad un modulo specializzato che analizza il modello di testo descritto nel capitolo 3 e, dall'analisi delle proposizioni e dei rispettivi verbi, crea il modello sotto forma di classi Java. Il prodotto dell'elaborazione consta di un'insieme di stati costituiti a loro volta da un insieme di variabili numeriche e da un insieme di equazioni e/o disequazioni. Tale modulo si ispira al progetto ARIS[4] dell'Allen Institute sia per quanto riguarda la classificazione dei verbi, sia per la modellazione del problema in stati. La differenza per quanto riguarda la classificazione dei verbi consiste nel metodo utilizzato: ARIS fa uso di algoritmi di apprendimento automatico, mentre in questo lavoro è stata usata una classificazione manuale affiancata dall'utilizzo di WordNet. Per quanto riguarda la modellazione, ARIS modella le entità in luoghi e applica transazioni di stato, mentre in questo modulo la rappre-

sentazione è più semplice e le entità sono modellate come variabili distinte dal nome ed appartenenti ad uno stato temporale.

La differenza di questo approccio rispetto a quello del capitolo 5 consta nel fatto che il tipo di problemi da risolvere sono espressi in maniera differente e richiedono un modello diverso per essere risolti. I problemi descrivono situazioni realistiche e richiedono il conteggio delle entità coinvolte. Non vengono espressi particolari vincoli logici e per la risoluzione è sufficiente calcolare semplici equazioni lineari.

La risoluzione del modello si svolge in più fasi e non implica il semplice calcolo delle operazioni matematiche, ma necessita anche di una elaborazione semantica per comprendere le relazioni tra le variabili ed una manipolazione dei diversi stati creati. Nel seguito sarà descritto nel dettaglio tutto il procedimento che è stato realizzato. In ausilio alla descrizione verranno usati i due esempi seguenti che, nella loro semplicità, catturano tutti gli aspetti più importanti di questa parte di lavoro.

Es. 6.0.1. "John has 5 cars. Mark has 3 truck. How many vehicles does Mark have? How many vehicles do they have total?"

Es. 6.0.2. "Jacob has 400 dollars, Lucy has 600 dollars and Frank has 320 dollars. Jacob gives 200 dollars to Lucy and Lucy send 25 dollars to Frank. How many dollars has Lucy?"

6.1 Costruzione del modello matematico

Il principio alla base della deduzione del modello matematico è l'analisi dei verbi delle proposizioni. L'interpretazione dei verbi si traduce in una o più equazioni o disequazioni che mettono in relazione le variabili derivanti dalle entità. Le variabili possono essere ricavate principalmente in due modi: traduzione di una singola entità nella rispettiva variabile o unione di due entità in un'unica variabile. Con il primo metodo è sufficiente usare come nome della variabile il nome dell'entità e come valore, se esiste, il quantificatore della relativa entità espresso dalla classe *Quantity*. Il secondo metodo è più complesso e verrà descritto nel seguito, in

linea generale il nome della variabile corrisponde alla concatenazione del nome delle due entità, tipicamente soggetto e complemento, ed il valore della variabile risultante corrisponde al quantificatore del complemento.

Lo scopo del problema lo si ricava dalla domanda del testo, in particolare lo scopo è costituito dalle variabili ed equazioni dedotte dalla domanda. La risoluzione, poi, partirà dall'obiettivo e cercherà di assegnare un valore alle variabili incognite.

Iniziamo col descrivere la semantica matematica dei verbi più semplici e più frequenti: il verbo ″essere″ ed il verbo ″avere″.

6.1.1 Il verbo essere ed avere

L'interpretazione del verbo ″essere″ ed ″avere″ sono più o meno equivalenti. La semantica di ″La gara ha meno di 30 partecipanti″ e ″I partecipanti alla gara sono meno di 30″ è esattamente la stessa e la sua traduzione in termini matematici è uguale. In particolare si può descrivere l'interpretazione del verbo essere come un'uguaglianza o disuguaglianza tra il soggetto ed il complemento oggetto. Nel seguito verrà usato il verbo avere per descrivere le traduzioni e, dove differente verrà illustrato il procedimento per il verbo essere.

La prima e più semplice traduzione del verbo ″avere″ corrisponde ad una equazione tra due variabili. Tale traduzione è sufficiente per creare il modello dell'esempio 6.0.1. Analizziamo quindi la prima proposizione: ″Jacob has 400 dollars″. Il procedimento consiste nel creare dapprima una variabile composta dal nome del soggetto più il nome del complemento assegnando un valore sconosciuto, in questo caso ″Jacob″ concatenato con ″dollar″ che dà come risultato `_Jacob_dollar`, poi un'altra variabile ricavata dal solo complemento oggetto con il valore uguale al rispettivo quantificatore, `400_dollar`, ed infine l'equazione che mette in relazione le due variabili:

$$\texttt{\_Jacob\_dollar} = \texttt{400\_dollar} \tag{6.1}$$

È stato scelto di usare sempre variabili e non semplici valori noti per la composizione delle equazioni, in questo modo i nomi delle variabili fun-

gono anche da unità di misura. Nell'equazione sopra c'è un'uguaglianza tra due entità di tipo "dollar", quindi il modello è coerente.

Nel caso in cui ci siano più soggetti l'interpretazione può variare a seconda degli avverbi presenti. Ad esempio: "Jacob and Lucy have two cars per each" viene tradotto in due equazioni, una per ogni soggetto:

$$\begin{aligned} \_\texttt{Jacob\_car} &= 2\_\texttt{car}, \\ \_\texttt{Lucy\_car} &= 2\_\texttt{car} \end{aligned} \tag{6.2}$$

Se non vi sono avverbi o viene specificato un avverbio che indichi "tutti insieme", ad esempio "All together Jacob and Lucy have 3 cars" viene creata un'unica equazione con le variabili dei soggetti sommate:

$$\_\texttt{Jacob\_car} + \_\texttt{Lucy\_car} = 3\_\texttt{car} \tag{6.3}$$

Nel caso vi siano più complementi oggetto, le equazioni saranno una per ogni complemento, per esempio "Jacob has one dog and two cats" diventa:

$$\begin{aligned} \_\texttt{Jacob\_dog} &= 1\_\texttt{dog}, \\ \_\texttt{Jacob\_cat} &= 2\_\texttt{cat} \end{aligned} \tag{6.4}$$

Mettendo insieme tutti questi casi è possibile ottenere diverse combinazioni, se ne riporta un ultimo esempio: "Jacob and Lucy have one dog and two cats per each", in questa frase ci sono due soggetti e due complementi, il rispettivo modello matematico sarà composto da quattro

equazioni.

$$\begin{aligned} \texttt{\_Jacob\_dog} &= 1\texttt{\_dog}, \\ \texttt{\_Jacob\_cat} &= 2\texttt{\_cat}, \\ \texttt{\_Lucy\_dog} &= 1\texttt{\_dog}, \\ \texttt{\_Lucy\_cat} &= 2\texttt{\_cat} \end{aligned} \tag{6.5}$$

Il verbo avere può produrre anche delle disequazioni. Frasi del tipo "The race has more/less/at least/at most than 20 partecipants" generano le rispettive disequazioni $\texttt{\_race\_partecipant} > 20\texttt{\_partecipant}$, $\texttt{\_race\_partecipant} < 20\texttt{\_partecipant}$, $\texttt{\_race\_partecipant} >= 20\texttt{\_partecipant}$ e $\texttt{\_race\_partecipant} <= 20\texttt{\_partecipant}$.

6.1.2 Il verbo iniziare

Il verbo "iniziare" inteso come ad indicare l'inizio di una numerazione, ovvero il limite inferiore del valore di una variabile, si aspetta un complemento introdotto dalla preposizione "from". La traduzione di una proposizione avente questo verbo si riflette nella creazione di una disequazione. Ad esempio la frase "Pectoral numbers start from 1" viene modellata come $\texttt{\_pectoral\_number} >= 1\texttt{\_pectoral\_number}$.

6.1.3 Lo stato

Per l'analisi dei verbi successivi è necessario introdurre il concetto di stato. Infatti è possibile distinguere due tipi di verbi: quelli che, come il verbo avere, descrivono una situazione statica e quelli che implicano un cambiamento di stato. Si pensi al verbo "dare", quando qualcuno detiene N oggetti di un qualche tipo e, ad un certo punto, ne dà m a qualcun altro, ciò comporta che la variabile che contava il numero di oggetti passa dal valore N al valore N - m. Per tenere traccia di tali cambiamenti è necessario modellare il concetto di stato.

Lo stato non è altro che l'insieme di tutte le variabili con i rispettivi valori. Quando si incontra un verbo che implica la modifica del valore di una o più variabili, lo stato viene duplicato e con esso tutte le variabili. Alle variabili interessate dalla modifica vengono ricalcolati i valori mentre le altre mantengono il valore precedente. I diversi stati sono messi in relazioni da equazioni che coinvolgono variabili dei differenti stati. I verbi che generano altri stati sono quelli che significano trasferire, aggiungere, rimuovere.

6.1.4 Il verbo trasferire

L'azione trasferire, muovere, spostare, dare etc. implica il trasferimento di una certa quantità da un certo punto, luogo, entità ad un altro. La quantità quindi è sottratta dal primo per essere aggiunta al secondo. A meno di avverbi che ne possano alterare il significato, in generale la traduzione di questa categoria di verbi si riflette in due equazioni, una per rimuovere una certa quantità e l'altra per aggiungerla.

Prendiamo ad esempio la proposizione "Jacob gives 200 dollars to Lucy" dell'esercizio 6.0.2. Come già descritto in precedenza, la variabile principale da generare, quella che subirà la modifica, coinvolge il soggetto ed il complemento oggetto, in questo caso "Jacob" concatenato a "dollars" ottenendo `_Jacob_dollar`. La quantità spostata corrisponde al complemento oggetto "200 dollars", mentre l'entità di destinazione coincide con il complemento preceduto dalla preposizione "to". Il risultato consiste nelle due equazioni seguenti:

$$\begin{aligned} &\texttt{\_Jacob\_dollar(2)} = 400\texttt{\_Jacob\_dollar(1)} - 200\texttt{\_dollar}, \\ &\texttt{\_Lucy\_dollar(2)} = 600\texttt{\_Lucy\_dollar(1)} + 200\texttt{\_dollar} \end{aligned} \tag{6.6}$$

Il numero tra parentesi indica l'indice dello stato a cui appartiene la variabile. Se una variabile era già stata definita precedentemente allora viene usata quella. In questo caso `_Jacob_dollar(1)` e `_Lucy_dollar(1)`

erano state definite dalle rispettive proposizioni ”Jacob has 400 dollars” e ”Lucy has 600 dollars”.

Varianti della traduzione principale sono causate, come in precedenza, dal numero di soggetti, dal numero di complementi e dagli avverbi. A seguire alcuni esempi con relative traduzioni.

Più soggetti come ”Jacob and Lucy give 5 dollars to Frank” generano:

$$\begin{aligned} &\_\texttt{Frank\_dollar(2)} = \_\texttt{Frank\_dollar(1)} + 5\_\texttt{dollar}, \\ &\_\texttt{Jacob\_dollar(2)} + \_\texttt{Lucy\_dollar(2)} = \\ &\_\texttt{Jacob\_dollar(2)} + \_\texttt{Lucy\_dollar(2)} - 5\_\texttt{dollar} \end{aligned} \tag{6.7}$$

Più complementi di termine, ad esempio ”Lucy gives 5 dollars to Frank and Jacob” si traducono in:

$$\begin{aligned} &\_\texttt{Lucy\_dollar(2)} = \_\texttt{Lucy\_dollar(1)} - 5\_\texttt{dollar}, \\ &\_\texttt{Jacob\_dollar(2)} + \_\texttt{Frank\_dollar(2)} = \\ &\_\texttt{Jacob\_dollar(2)} + \_\texttt{Frank\_dollar(2)} + 5\_\texttt{dollar} \end{aligned} \tag{6.8}$$

Gli avverbi modificano la traduzione aumentando il numero di equazioni. Per esempio ”Lucy gives 5 dollars to each of them”, con ”them” riferito a Jacob e Frank:

$$\begin{aligned} &\_\texttt{Lucy\_dollar(2)} = \_\texttt{Lucy\_dollar(1)} - 5\_\texttt{dollar}, \\ &\_\texttt{Frank\_dollar(2)} = \_\texttt{Frank\_dollar(1)} + 5\_\texttt{dollar}, \\ &\_\texttt{Jacob\_dollar(2)} = \_\texttt{Jacob\_dollar(1)} + 5\_\texttt{dollar} \end{aligned} \tag{6.9}$$

Infine ”Each of them give 5 dollars to Lucy”, con ”them” riferito sem-

pre a Jacob e Frank, produce:

$$\begin{aligned}
&\texttt{_Frank_dollar(2)} = \texttt{_Frank_dollar(1)} - \texttt{5_dollar},\\
&\texttt{_Jacob_dollar(2)} = \texttt{_Jacob_dollar(1)} - \texttt{5_dollar},\\
&\texttt{_Lucy_dollar(2)} = \texttt{_Lucy_dollar(1)} + \texttt{5_dollar} + \texttt{5_dollar}
\end{aligned} \tag{6.10}$$

In quest'ultimo esempio si noti che ai soldi di Lucy deve essere aggiunta la quantità tante volte quanti sono i soggetti che cedono la quantità stessa.

6.2 Risoluzione

La risoluzione del modello matematico implementata consta in più passi ed itera sul modello più volte. Si possono individuare tre operazioni principali che vengono svolte al fine di trovare la soluzione: risoluzione semantica delle variabili, risoluzione delle singole equazioni e propagazione dei nuovi valori nei differenti stati.

Per la descrizione della risoluzione semantica delle variabili serviamoci dell'esempio 6.0.1. Dalle prime due frasi si ricavano le due variabili `5_John_car` e `3_Mark_truck`. Dalla prima domanda si ricava la variabile incognita `_Mark_vehicle`. Si noti che nella domanda "Mark" è il soggetto e "vehicles" il complemento oggetto, le due parti logiche sono invertite rispetto alla normale forma di frase poiché la domanda ha una struttura differente. La costruzione della variabile è comunque coerente rispetto a quanto esposto sopra, ovvero è sempre formata dalla concatenazione del soggetto seguito dal complemento.

La risoluzione del problema parte sempre dalle variabili o dalle equazioni ricavate dalla domanda. La risoluzione semantica delle variabili, in questo caso, inizia cercando tra tutte le variabili create quella che può avere una certa correlazione di significato con `_Mark_vehicle`. In particolare viene usato il metodo *float hypernymDistance(String word, String hypernym, POS pos)* dell'interfaccia *ISemantic* per valutare se i nomi o parti dei nomi

delle variabili sono in relazione di iperonimia. Nel caso specifico, quando viene confrontata la variabile `5_John_car` con `_Mark_vehicle` il risultato è che ”vehicle” è un sostantivo di significato più generico di ”car”, mentre ”Mark” e ”John” non sono correlati. Quando si prova il confronto tra `5_Mark_truck` e `_Mark_vehicle`, ”vehicle” e ”truck” sono in relazione di iperonimia e ”Mark” e ”Mark” sono uguali, quindi alla variabile `_Mark_vehicle` viene assegnato il valore della variabile `5_Mark_truck` ovvero 5. La prima soluzione al problema è già trovata. Dalla seconda e ultima domanda, ricordando che ”they” viene sostituito da ”John and Mark”, si trova invece la seguente equazione:

$$\texttt{\_John\_vehicle} + \texttt{\_Mark\_vehicle} = \texttt{\_vehicle} \tag{6.11}$$

L’equazione presenta tre variabili, tutte e tre incognite, si procede allora alla risoluzione semantica prima con la variabile `_John_vehicle` che trova la corrispondenza con `_John_car` e poi con la variabile `_Mark_vehicle`. A questo punto l’equazione presenta una sola incognita e può essere risolta per via analitica trovando la soluzione richiesta: $\texttt{\_vehicle} = 8$.

Per la risoluzione delle equazioni è stato implementato un componente specifico che è in grado di risolvere singole equazioni ad una sola incognita. Questo risolutore analizza, tramite il *pattern visitor*, il modello Java dell’equazione, esegue le operazioni matematiche tra le variabili ed assegna il valore calcolato alla variabile incognita. Il modello Java dell’equazione è costituito da due espressioni, quella a sinistra dell’uguale e quella a destra. Un’espressione è definita come una sommatoria di termini. I termini possono essere dei termini prodotto o frazione, espressioni o delle semplici variabili numeriche. Il diagramma UML 6.1 illustra il modello appena descritto.

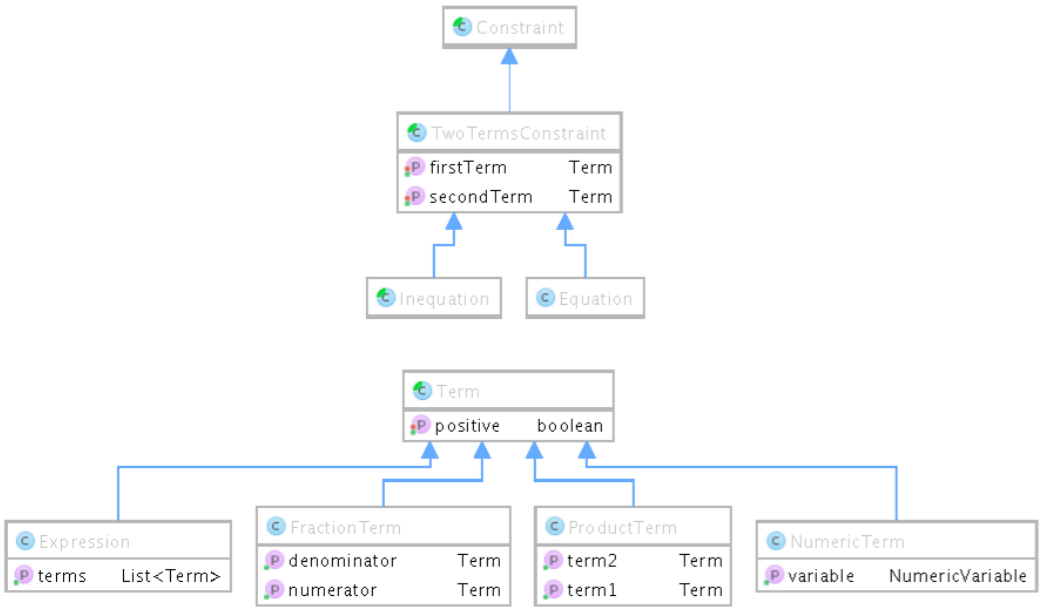


Figura 6.1: La classe *Equation*

L'ultimo procedimento da descrivere riguarda la propagazione dei nuovi valori calcolati tra i diversi stati. Gli stati contengono ognuno una copia di tutte le variabili e, quando ad una incognita viene assegnato un valore, è necessario propagare la modifica agli altri stati. L'esempio 6.0.2 necessita di questo meccanismo per essere risolto, infatti, come si può dedurre dal testo, ci sono due spostamenti di denaro, prima da Jacob a Lucy, poi da Lucy a Frank. Il trasferimento implica che una stessa variabile assumerà

diversi valori nel tempo. Di seguito il modello completo del problema:

$$
\begin{aligned}
&\texttt{\_Jacob\_dollar(1)} = 400\texttt{\_dollar},\\
&\texttt{\_Lucy\_dollar(1)} = 600\texttt{\_dollar},\\
&\texttt{\_Frank\_dollar(1)} = 320\texttt{\_dollar},\\
&\texttt{\_Jacob\_dollar(2)} = 400\texttt{\_Jacob\_dollar(1)} - 200\texttt{\_dollar},\\
&\texttt{\_Lucy\_dollar(2)} = 600\texttt{\_Lucy\_dollar(1)} + 200\texttt{\_dollar},\\
&\texttt{\_Lucy\_dollar(3)} = \texttt{\_Lucy\_dollar(2)} - 25\texttt{\_dollar},\\
&\texttt{\_Frank\_dollar(3)} = \texttt{\_Frank\_dollar(2)} + 25\texttt{\_dollar}
\end{aligned}
\tag{6.12}
$$

Poiché sono presenti due trasferimenti, gli stati saranno tre. Il primo trasferimento lega lo stato iniziale con lo stato successivo ed il secondo trasferimento lega il secondo stato al terzo. Dato che al momento della creazione del modello non c'è nessun tipo di risoluzione, le variabili coinvolte nelle equazioni avranno un valore sconosciuto negli stati successivi. In figura 6.2 come si presentano gli stati non appena il modello è stato creato. Nello stato 2 `_Jacob_dollar` e `_Lucy_dollar` sono ignote perché compaiono nella quarta e quinta equazione del modello sopra. Nello stato 3 anche `_Frank_dollar` è incognita perché il suo valore cambia a causa dell'ultima equazione del modello.

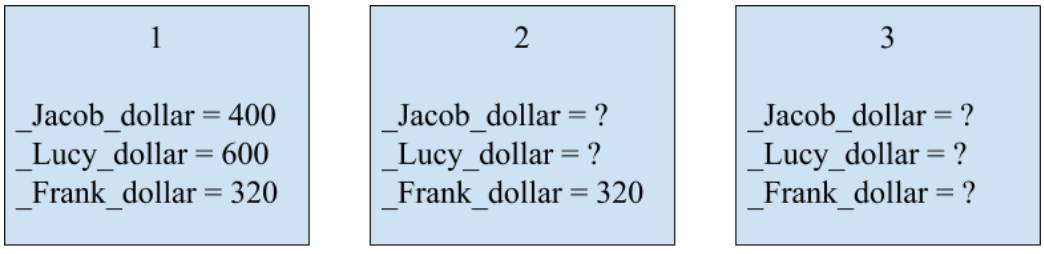


Figura 6.2: Gli stati non appena creato il modello

Non appena vengono calcolate la quarta e la quinta equazione, il nuo-

vo valore della variabile `_Jacob_dollar` viene propagato sia allo stato 2 che allo stato 3, mentre il valore della variabile `_Lucy_dollar` viene aggiornato solamente allo stato 2 poiché la variabile è nuovamente coinvolta nell'equazione che lega questo stato al terzo.

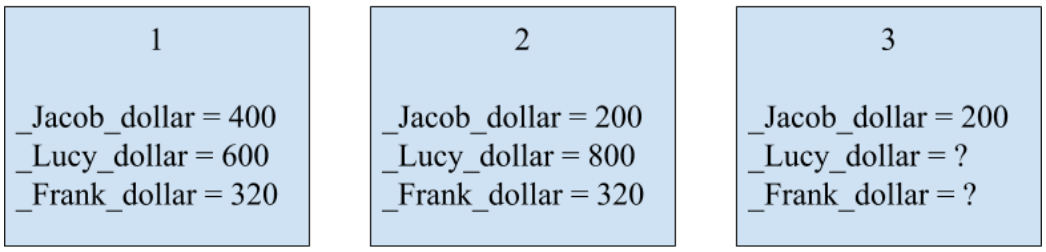


Figura 6.3: Gli stati dopo la risoluzione della quarta e quinta equazione del modello

Appena tutte le equazioni sono risolte gli stati contengono tutti i valori calcolati.

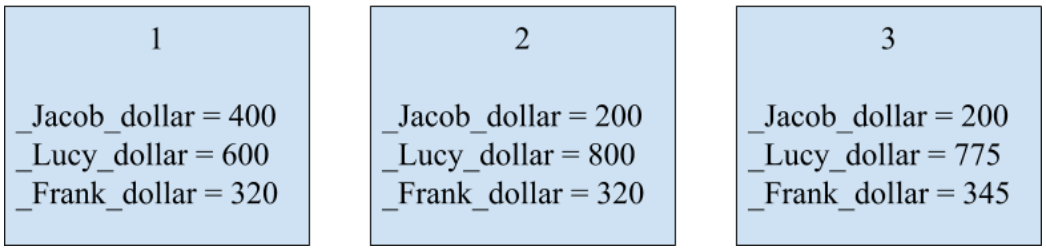


Figura 6.4: Gli stati alla fine della risoluzione

Con l'ausilio di questo modello a stati è possibile selezionare le variabili anche nello spazio temporale: se il problema chiedesse i valori iniziali o finali è sufficiente mostrare le variabili dello stato iniziale o finale rispettivamente.

6.3 Conclusioni

Tramite questi strumenti realizzati è possibile modellare semplici problemi algebrici in sistemi di equazioni lineari. Il sistema potrebbe essere

potenziato in diversi modi. Aggiungere interpretazioni alle categorie di verbi che ancora non ce l'hanno aumenterebbe il numero di problemi risolvibili e migliore sarà la scelta dei verbi a cui dare semantica maggiore sarà la copertura finale sui quesiti. Un altro approccio interessante potrebbe essere quello di estendere il metodo per la creazione del modello ispirandosi ad esempio al lavoro [5] quindi usando dei modelli predefiniti di modo da poter dedurre più facilmente dei sistemi di equazione più complessi in grado di risolvere anche problemi non esclusivamente matematici, per esempio, fisici.

Capitolo 7

Primi risultati sperimentali

Il sistema sistema software realizzato è stato sviluppato seguendo due strade: una per la risoluzione di problemi logico-matematici e l'altra per la risoluzione di problemi aritmetici. Di conseguenza per valutare le prestazioni è necessario testare separatamente i due moduli su due insiemi di esercizi distinti in base alla loro tipologia. In questo capitolo, oltre a mostrare i risultati in termini numerici, verranno discussi le limitazioni della presente soluzione e, nelle conclusioni, le possibili alternative per migliorare il sistema. I criteri di valutazione si basano sul concetto di copertura e precisione. Sia k il numero di quesiti ai quali il sistema fornisce la risposta corretta, la copertura è definita come il rapporto tra k ed il numero totale di problemi n, mentre la precisione è definita come k sul numero dei problemi m a cui il sistema dà una risposta.

$$\begin{aligned} copertura &= k/n \\ precisione &= k/m \end{aligned} \tag{7.1}$$

7.1 Problemi logico-matematici

Per la valutazione del primo modulo, quello che si occupa della risoluzione di problemi logico-matematici, è stata utilizzata una parte della collezione di quesiti raccolta dalla University of Science and Technology

of China per testare il loro sistema Dolphin[2]. Il database[1] è a sua volta costituito da un sottoinsieme di problemi reperibili da algebra.com e da Yahoo Answer[2]. Questo database è costituito da problemi che, per essere risolti, necessitano quasi esclusivamente di una conoscenza matematica, sono esclusi invece quelli che comportano una certa conoscenza di base sul mondo reale. Gli esercizi presenti sono inoltre classificabili in altre sotto-tipologie, o meglio, studiandone il testo e la rispettiva soluzione si possono riconoscere degli schemi ricorrenti, infatti spesso i quesiti sono uguali e variano soltanto per le quantità in gioco o per qualche alterazione a livello sintattico. Ne riportiamo di seguito uno a titolo di esempio:

> "The sum of two numbers is 64. The smaller number is 12 less than the larger number. What are the two numbers?"

Con questo campione di esercizi, il modulo in esame, su un insieme di 150 esercizi, presenta una copertura pari al 33% mentre la precisione si attesta al 70%. Un terzo dei problemi viene risolto in modo corretto e su tutti i risultati che genera il sistema, il 70% è corretto. I risultati mostrano i limiti della soluzione attuale, ma analizziamo per quali motivi la risoluzione fallisce.

I fallimenti sono dovuti per la gran parte alla mancanza di interpretazione di alcuni concetti matematici, mentre altri per una mancata risoluzione di riferimenti nel testo. Nello specifico le nozioni matematiche di cui non è stata sviluppata un'interpretazione sono i concetti di numero consecutivo, il concetto di numero opposto e l'analisi di formule matematiche scritte esplicitamente nel testo. I quesiti che contengono, ad esempio, un'espressione del tipo "il più grande dei numeri pari consecutivi" viene risolto, ma, mancando il concetto di numero consecutivo, l'interpretazione valuta la frase come fosse "il più grande dei numeri pari" e quindi risponde sì, ma in modo errato.

Altri problemi sono legati più ad un fattore di analisi del testo. Per esempio se un problema recita "The second of two numbers is 4 more than

[1] http://research.microsoft.com/en-us/projects/dolphin/
[2] https://answers.yahoo.com/

the first. The sum of the numbers is 56. If x represents the first number, and y represents the second number, what are the numbers?", il risolutore non è in grado di capire quale sia il primo ed il secondo numero a cui ci si riferisce. Un altro esempio simile è il seguente "The sum of two numbers is 2. If one number is subtracted from the other, the result is 8. Find the numbers.", anche in questo caso il sistema non sa interpretare "one number" e "the other" oltre a non capire a cosa si riferisca "the result".

7.2 Problemi aritmetici

Per quanto riguarda i problemi aritmetici, obiettivo del secondo risolutore, è stato usato il database[3] fornito dall'Allen Institute ed utilizzato per il loro sistema ARIS[4] che riesce ad ottenere una copertura del 77,7%. I quesiti sono tutti abbastanza simili e variano principalmente per i verbi utilizzati. Un esempio può essere il seguente:

> "Joan found 70 seashells on the beach. She gave Sam some of her seashells. She has 27 seashell. How many seashells did she give to Sam?"

Il risultato del test evidenzia una copertura pari al 47% ed una precisione del 73% su 100 problemi. I risultati sono incoraggianti, ma analizziamo quali sono stati gli errori. In questo caso le cause per le quali il sistema non riesce a rispondere consistono nella mancanza dell'interpretazione di alcuni verbi. Nonostante siano stati implementati algoritmi allo scopo di generalizzare il più possibile il significato dei verbi è evidente che è necessario aumentare la copertura delle categorie di verbi aggiungendone altri. Invece, esercizi ai quali il sistema risponde con risultato non corretto riguardano problemi che distinguono le entità per i loro aggettivi, ad esempio "Jason has 43 blue and 16 red marbles. Tom has 24 blue marbles. How many blue marbles do they have in all?", il risolutore, come spiegato nel capitolo 6, crea le variabili associando solo il nome delle entità e non i rispettivi aggettivi, di conseguenza non è possibile distinguere le variabili.

[3] https://www.cs.washington.edu/nlp/arithmetic

Un'ulteriore difficoltà riscontrata risiede nell'analisi delle domande: quando queste sono sintatticamente più articolate ovvero la domanda non è semplice e diretta, ma mescolata con informazioni aggiuntive sul problema, il risolutore non è in grado di capire cosa viene richiesto.

7.3 Conclusioni

Dalle analisi degli errori su entrambi i moduli si deduce che la maggior parte dei fallimenti è dovuta alla mancata implementazione di determinate interpretazioni. Per il primo componente e rispettiva tipologia di quesiti è più importante attribuire significato matematico ai sostantivi e relativi aggettivi. Infatti, molto spesso gli unici verbi utilizzati sono il verbo "essere" e il verbo "avere" con uguale semantica, è il sostantivo che definisce gran parte del problema. Ad esempio "La *somma* di *due numeri* è uguale al *doppio* della loro *differenza*", il verbo implica semplicemente l'uguaglianza tra due entità, ma sono le entità ad avere un'interpretazione complessa e significativa.

Per quanto riguarda il secondo componente, invece, è necessario implementare più interpretazioni dei verbi o includere altri verbi ad interpretazioni già esistenti. Il problema riguardante la non considerazione degli aggettivi per comporre il nome delle variabili è di più lieve entità e può essere risolto più facilmente.

Nonostante i numeri premino il secondo modulo, c'è da considerare che i problemi che risolve il primo componente sono concettualmente più difficili ed interessanti, nonché la sua estensione più facile e con maggiori potenzialità. In riferimento a quanto spiegato sopra, già implementando il concetto di numero consecutivo, secondo il database di quesiti considerato, la percentuale di problemi che sarebbe in grado di risolvere il primo modulo aumenterebbe di molto: circa il 40% dei problemi non risolti o risolti in modo sbagliato includono il suddetto concetto matematico.

Dato il lavoro svolto, l'idea di aggiungere le interpretazioni mancanti è la soluzione che viene più spontanea, ma da una riflessione più accurata si potrebbe optare per vie alternative. Si consideri che continuando su que-

sta strada è vero che si aumenterebbero i quesiti risolvibili, ma non appena si presenti una variazione non prevista sul problema il sistema sbaglierebbe. L'uso di una metodologia di apprendimento automatico da affiancare agli attuali algoritmi renderebbe il sistema più flessibile e aumenterebbe la comprensione del testo più articolato, il tutto ad un costo implementativo minore rispetto all'implementazione manuale. Nella fattispecie sarebbe interessante sperimentare un algoritmo di apprendimento che impari ad associare dei modelli predefiniti a parti di frase o a frasi intere e, tramite algoritmi come quelli sviluppati in questo progetto di tesi, procedere con un completamento dei suddetti modelli in modo preciso.

Conclusioni

Il lavoro che è stato fatto per questo progetto di tesi si è concentrato in particolar modo su due aspetti: elaborazione del linguaggio naturale e deduzione di un modello matematico. Per la prima parte si sono studiati e sperimentati diversi strumenti al fine di riuscire ad analizzare al meglio il testo in linguaggio naturale. Dallo studio fatto è emerso che lo stato degli strumenti a disposizione per NLP non è ancora maturo, infatti a volte sbagliano ad interpretare parti di testo. Si sono riscontrati problemi nell'analisi grammaticale, ad esempio errori nella classificazione tra avverbio e aggettivo, e incapacità di risolvere i riferimenti espressi dai pronomi. Come risultato si ha che il testo viene compreso se questo rimane semplice a livello di struttura di frase altrimenti si incorre in imprecisioni, mancanze o errori che compromettono le fasi successive.

Per quanto riguarda la deduzione del modello matematico sono stati realizzati due moduli distinti: uno adatto a risolvere problemi logico-matematici l'altro dedicato a quesiti di tipo aritmetico. I principi guida di entrambi i componenti sono stati l'interpretazione precisa dei verbi e delle entità. La differenza sostanziale tra i due consiste nel fatto che il primo crea un modello logico scritto in Prolog ed il secondo crea un modello composto da equazioni e disequazioni supportato da una rappresentazione a stati.

Dal lavoro svolto è emerso che il primo modulo, quello che produce in uscita una rappresentazione Prolog, anche se dai test è risultato più scadente, è più potente nella modellazione, più facile da implementare ed estendere grazie alla corrispondenza verbo-predicato ed entità-predicato. In futuro è auspicabile far convergere i due moduli sviluppati in uno uni-

co. Infatti i due componenti condividono alcune procedure per la costruzione del modello e possono convergere creando un'unica rappresentazione finale descritta in un linguaggio logico a vincoli.

Una ulteriore estensione potrebbe essere fatta facendo uso di algoritmi di apprendimento automatico che imparino a mappare tipi di frase a specifici "template" di vincoli. I template ottenuti possono essere poi completati tramite un altro algoritmo che ne mappi le variabili creando una precisa corrispondeza con gli elementi del testo. Il suddetto approccio si ispira al progetto di ricerca, descritto nella pubblicazione [3], basato sulla deduzione di modelli predefiniti da forme di frasi in linguaggio naturale. La differenza consisterebbe nel mappare, invece che modelli di equazioni, modelli di vincoli. Tale approccio limiterebbe drasticamente l'implementazione manuale delle interpretazioni ed aumenterebbe la flessibilità nonché la copertura delle frasi comprensibili.

Per quanto riguarda la comprensione semantica del testo, al momento il sistema realizzato riesce a comprendere relazioni di similarità basate principalmente sul concetto di iperonimia e sinonimia. Ciò implica una buona flessibilità in caso il testo usi sinonimi o comunque termini differenti per intendere significati simili. Per quanto riguarda concetti collegati da relazioni semantiche più complesse, basate su una conoscenza di base come ad esempio la capacità di dedurre che se i partecipanti ad una gara sono *N*, di conseguenza anche i pettorali sono *N*, il sistema attualmente non è in grado di fare tali ragionamenti. Il sistema software sviluppato in questo lavoro di tesi, nonostante non possegga una semantica sulla situazione reale in cui è calato il problema matematico, riesce comunque a risolvere quesiti che non siano espressi in termini solo matematici.

Rispetto agli altri progetti esistenti che condividono con questo lavoro l'obiettivo di risolvere problemi matematici espressi in linguaggio naturale, questa realizzazione si è concentrata ed è in grado di risolvere una tipologia di problemi differente: problemi di tipo logico-matematico risolti da una rappresentazione logica quale Prolog.

Nel futuro, implementando quanto detto sopra ovvero miglioramento dell'analisi del linguaggio naturale, uso di algoritmi di apprendimento

automatico e estensione del modello dedotto in un modello a vincoli, sarà possibile avere un sistema molto potente in grado di risolvere un maggiore numero di problemi matematici.

Bibliografia

[1] Sarah Witzig, *Accessing WordNet from Prolog*, Artificial Intelligence Center, The University of Georgia, 8 Maggio 2003.

[2] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu e Yong Rui, *Automatically Solving Number Word Problems by Semantic Parsing and Reasoning*, Microsoft Research, University of Science and Technology of China.

[3] Nate Kushmany, Yoav Artziz, Luke Zettlemoyerz e Regina Barzilay, *Learning to Automatically Solve AlgebraWord Problems*, Computer Science and Articial Intelligence Laboratory, Massachusetts Institute of Technology, Computer Science & Engineering, University of Washington.

[4] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni e Nate Kushman, *Learning to Solve ArithmeticWord Problems with Verb Categorization*, University of Washington, Allen Institute for AI, Massachusetts Institute of Technology.

[5] Kyle Morton e Yanzhen Qu, *A Novel Framework for Math Word Problem Solving*, February 2013.

[6] Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey e Thierry Petit, *Global Constraint Catalogue: Past, Present and Future*, February 2007.

[7] Patrick Blackburn e Johan Bos, *Computational Semantics for Natural Language*, Indiana University, June 17-21, 2003.

[8] Aldo Gangemi, Claudio Sacerdoti Coen, Alexandre Passant e Aldo Gangemi *Ontologies and Languages for Representing Mathematical Knowledge on the Semantic Web*.

[9] Iddo Lev, *PACKED COMPUTATION OF EXACT MEANING REPRESENTATIONS*, June 2007.

[10] Subhro Roy, Tim Vieira e Dan Roth, *Reasoning about Quantities in Natural Language*.

[11] Iddo Lev, Bill MacCartney, Christopher D. Manning e Roger Levy, *Solving Logic Puzzles: From Robust Processing to Precise Semantics*, Stanford University.

[12] http://matematica.unibocconi.it/giochi-matematici

[13] https://opennlp.apache.org/

[14] http://nlp.stanford.edu/

[15] https://www.wiktionary.org/

[16] https://dkpro.github.io/dkpro-jwktl/

[17] https://wordnet.princeton.edu/

[18] http://www.swi-prolog.org/

Elenco delle figure