Report sulla attività progettuale di
Fondamenti di Intelligenza Artificiale **Michele Di Capua – MAT 0001006103**

Natural Language Processing in Prolog

Lo scopo principale del progetto è stato quello di rendere il programma *Talk* presentato nel libro "*Prolog and Natural-Language Analysis*" di F. C. N. Pereira e S. M. Shieber pienamente integrato con *WordNet*, un database lessicale elaborato da G. A. Miller presso l'Università di Princeton.

Il punto di partenza del progetto è stato un analogo lavoro svolto dallo studente Lorenzo Rosa, che ha prodotto un programma di nome *MyTalk* in grado di interfacciarsi con *WordNet*, sebbene in versione limitata soprattutto per quanto riguarda i verbi. In questo lavoro si è mantenuta la struttura generale del programma *MyTalk*, semplificandolo in alcune parti e apportando le opportune modifiche per farlo interfacciare con più istanze di *WordNet*.

Il Capitolo 1 di questo documento offre una panoramica sintetica e generale del contesto all'interno del quale il progetto è inserito per fornire quegli elementi che paiono necessari per comprenderne a pieno il senso.

Il Capitolo 2 mostra quelli che sono i problemi evidenziati nel programma *MyTalk* e che non permettono di raggiungere a pieno l'obiettivo di un *Talk* effettivamente interattivo con *WordNet*.

Il Capitolo 3 mostra le principali modifiche fatte al programma per renderlo funzionante.

Il Capitolo 4 propone a titolo di esempio alcune interazioni con il programma finale, chiamato *NotMyTalk*, paragonate con l'equivalente in *MyTalk*, mostrando qualitativamente i miglioramenti ottenuti con le modifiche effettuate.

Il Capitolo 5 individua in breve gli eventuali sviluppi futuri del programma.

Report sulla attività progettuale di
Fondamenti di Intelligenza Artificiale **Michele Di Capua – MAT 0001006103**

1. IL CONTESTO

Il Natural Language Processing (NLP) è un insieme di tecniche per l'analisi e la rappresentazione automatica del linguaggio naturale tramite calcolatore. Obiettivo di questa disciplina è quello di raggiungere un'elaborazione del linguaggio simile a quella umana, per questa ragione è considerata una branchia dell'Intelligenza Artificiale (AI) nonostante sia connessa anche a molte altre discipline. Data l'ampiezza dell'obiettivo, è possibile individuare all'interno dell'NLP più applicazioni, ciascuna con un proprio task specifico. Le più comuni sono:

1) Information Retrieval: si occupa di recuperare informazioni fornendo in risposta alla domanda di un utente un elenco di documenti potenzialmente rilevanti. È il campo dei motori di ricerca.
2) Information Extraction: data una rappresentazione strutturata, si concentra sul riconoscimento, il tagging e l'estrazione di elementi chiave di informazione quali ad esempio persone, aziende, luoghi ecc…
3) Question-Answering: al contrario dell'Information Retrieval, punta a fornire all'utente una risposta il più esatta possibile sfruttando una base di conoscenza.
4) Summarization: mira a ridurre una grande quantità di testo in un testo più corto, mantenendo significato e informazioni fondamentali del testo originale.
5) Machine Translation: studia la traduzione automatica di testi da una lingua ad un'altra.
6) Dialogue Systems: è focalizzato sulla creazione di sistemi in grado di reggere dialoghi con gli umani, solitamente legato ad un'applicazione mirata.

Il modo più esplicativo per presentare ciò che accade all'interno di un sistema per l'NLP è attraverso il modello dei "livelli del linguaggio".

1) Livello fonologico: riguarda l'interpretazione dei suoni e la loro traduzione in parole.
2) Livello morfologico: riguarda una analisi sulle componenti naturali delle parole, quali prefissi, radici e suffissi. È fondamentale, ad esempio, per il riconoscimento dei verbi e dei loro tempi.
3) Livello lessicale: è legato all'interpretazione delle singole parole.
4) Livello sintattico: si concentra sul ruolo delle parole all'interno di una frase, in modo da individuarne la struttura grammaticale.
5) Livello semantico: determina il possibile significato di una frase indagando le relazioni tra i significati delle singole parole nella frase. Si occupa, ad esempio, di dipanare le eventuali disambiguità semantiche nella frase.

Un'ulteriore precisazione va fatta su quelli che sono gli approcci principali all'NLP:

1) Approccio simbolico: si basa sulla rappresentazione esplicita della struttura simbolica del linguaggio. Nei sistemi logici, tale struttura è resa attraverso proposizioni logiche.
2) Approccio statistico: implementa varie tecniche matematiche spesso applicate a una grande quantità di testo per sviluppare dei modelli generali del linguaggio. Si basa dunque sull'uso di dati osservabili come fonte primaria, traendo da questi i modelli necessari per l'analisi e l'elaborazione del linguaggio in studio.

Noto ciò è possibile affermare che il progetto in questione si avvale di un approccio simbolico all'NLP, utilizzando infatti il PROLOG, e opera principalmente sui livelli morfologico, lessicale e sintattico del linguaggio. Un'ultima osservazione può essere fatta sul programma Talk che è possibile ricondurre alla branchia del Query-Answering. Questo infatti si occupa di prendere in input frasi che possono essere affermazioni o domande: nel caso di affermazioni, le trasforma in proposizioni logiche del primo ordine creando la propria "base di conoscenza"; nel caso di domande, indaga all'interno della propria base di conoscenza per fornire una risposta all'utente.

2. PROBLEMI EVIDENZIATI

Effettuando alcune prove è possibile notare come in *MyTalk* sono solo alcuni i verbi ammessi di tutti quelli presenti in *WordNet*. Si riportando di seguito alcuni esempi esplicativi:

```
>> I wrote songs
Asserted "writes(I,song)."
>> every engineer reflects
Asserted "reflect(_29858):-engineer(_29858)."
>> he sings songs
Asserted "sings(he,song)."
>> he likes dogs
Error: "too difficult."
>> mike walks
Error: "too difficult."
>> terry plays football
Error: "too difficult."
```

Ciò che si è notato è che vi è un errore nell'uso della libreria *wn_fr.pl*. L'errore pare derivare dalla documentazione fornita dal sito dedicato ai tool di ProNTo. In particolare, nel file "Accessing WordNet from Prolog." (Witzig Sarah, 2003), consultabile al link http://ai1.ai.uga.edu/mc/pronto/Witzig.pdf , è scritto che la struttura dei predicati nel file è la seguente:

```
fr(synset_ID, f_num, w_num).
```

Dove `synset_ID` indica il *synset* al quale il verbo appartiene, `f_num` indica il *sentence frame*, mentre `w_num` indica la parola all'interno del *synset* a cui si riferisce il predicato, oppure nel caso il suo valore sia 0, indica che il predicato si riferisce all'intero *synset*.

Sulla base di ciò il programma *MyTalk* utilizza il file *wn_fr.pl* per la generazione dei verbi. In particolare si osservino le righe dalla 653 alla 657 del codice:

```
iv( no, [[W, -s]], [[W, -ed]], [[W, -past]],
      [[W, -ed]], [[W, -en]],
      [[W, -past]], [[W, -ing]], FOL, sg):- s(SysID,W_Num,W,v,_,_),
                                            fr(SysID, FR, W_Num),
                                            sen_fol_iv(FR, W, FOL).
```

Il programma recupera il *sentence frame* dal predicato `fr(SysID, FR, W_Num)` e lo utilizza in `sen_fol_iv(FR,W,FOL)` per individuare l'asserzione che ci fornisce la FOL del verbo tra quelle codificate nel file *sen_fol.pl*.

Analizzando il file *wn_fr.pl* ci si accorge però come la struttura dei predicati non sia quella indicata dal documento precedentemente citato, e dunque quella utilizzata nel codice di MyTalk. Si riportano a titolo di esempio le prime 5 righe del file:

```
fr(200001740,0,2).
fr(200001740,0,8).
fr(200002325,0,2).
fr(200002573,0,2).
fr(200002724,0,2).
```

Appare evidente come la struttura sia:

```
fr(synset_ID,w_num,f_num),
```

e non

```
fr(synset_ID,f_num,w_num).
```

3. RISOLUZIONE

Applichiamo dunque le seguenti modifiche al codice nelle righe suddette e in tutte le simili occorrenze successive:

```
iv( no, [[W, -s]], [[W, -ed]], [[W, -past]],
      [[W, -ed]], [[W, -en]],
      [[W, -past]], [[W, -ing]], FOL, sg):- s(SysID,_,W,v,_,_),
                                                fr(SysID, _, FR),
                                                sen_fol_iv(FR, W, FOL).
iv( [[W]], no, [[W, -ed]], [[W, -past]],
      [[W, -ed]], [[W, -en]],
       [[W, -past]], [[W, -ing]], FOL, pl):- s(SysID,_,W,v,_,_),
                                                 fr(SysID, _, FR),
                                                 sen_fol_iv(FR, W, FOL).



tv( no, [[W, -s]], [[W, -ed]], [[W, -past]],
      [[W, -ed]], [[W, -en]],
      [[W, -past]], [[W, -ing]], FOL, sg):- s(SysID,_,W,v,_,_),
                                                fr(SysID, _, FR),
                                                sen_fol_tv(FR, W, FOL).

tv( [[W]], no, [[W, -ed]], [[W, -past]],
      [[W, -ed]], [[W, -en]],
      [[W, -past]], [[W, -ing]], FOL, pl):- s(SysID,_,W,v,_,_),
                                                fr(SysID, _, FR),
                                                sen_fol_tv(FR, W   FOL).
```

Si ottiene così un *Talk* in grado di interagire correttamente con i verbi di wordnet.

4. RISULTATI FINALI

Il programma *NotMyTalk* è ora in grado di interagire con un dizionario di verbi più ampio. Si riportano di seguito alcuni esempi tratti da *NotMyTalk* con accanto i rispettivi risultati di *MyTalk*.

NotMyTalk	MyTalk

`>> mike is an engineer`
`Asserted "engineer(mike)."`
 	
`>> every engineer loves maths`
`Asserted "loves(_21816,math):-`
`engineer(_21816)."`
 	
`>> terry is an architect`
`Asserted "architect(terry)."`
 	
`>> every architect hates maths`
`Asserted "hates(_23152,math):-`
`architect(_23152)."`
 	
`>> who hates maths`
`terry.`
 	
`>> who loves maths`
`mike.`
	`>> mike is an engineer`
	`Error: "too difficult."`
	
	`>> every engineer loves maths`
	`Error: "too difficult."`
	
	`>> terry is an architect`
	`Error: "too difficult."`
	
	`>> every architect hates maths`
	`Error: "too difficult."`
	
	`>> who hates maths`
	`Error: "too difficult."`
	
	`>> who loves maths`
	`Error: "too difficult."`

`>> he runs` `Asserted "run(he)."`	`>> he runs` `Error: "too difficult."`
`>> he likes cooking` `Asserted "likes(he,cooking).`	`>> he likes cooking` `Error: "too difficult."`

`>> pizza is a food`
`Asserted "food(pizza)."`
 	
`>> dogs eat every food`
`Asserted "eats(dog,_2956):-`
`food(_2956)."`
 	
`>> who eats pizza`
`dog.`
	`>> pizza is a food`
	`Asserted "food(pizza)."`
	
	`>> dogs eat every food`
	`Error: "too difficult."`
	
	`>> who eats pizza`
	`Error: "too difficult."`

5. SVILUPPI FUTURI

Per rendere il programma *NotMyTalk* ancora più funzionale uno dei passi necessari è mettere mano alla struttura delle frasi. Ad ora, infatti, sono permesse solo frasi composte da soggeto-predicato-complemento oggetto, con la sola aggiunta di un quantificatore universale. Dare la possibilità di analizzare anche frasi più complesse in struttura o frasi che abbiano al loro interno anche aggettivi/avverbi aumenterebbe di molto le potenzialità del programma.

BIBLIOGRAFIA

Blackburn, P., & Bos, J. (2005). Representation and inference for natural language.

C. Matthews (1998). An Introduction to Natural Language Processing Through Prolog.

Miller, G. A. (1995). WordNet: a lexical database for English.

MyTalk e materiale legato all'attività progettuale di Lorenzo Rosa. https://github.com/ellerre/MyTalk

Pereira, F. C., & Shieber, S. M. (1987). Prolog and natural-language analysis.

ProNTo – Prolog Natural Language Tools. http://ai1.ai.uga.edu/mc/pronto/

Ribeiro, C. et al. (2004). INQUER: A WordNet-based Question-Answering Application.

Witzig, Sarah (2003) - Accessing WordNet from Prolog. http://ai1.ai.uga.edu/mc/pronto/Witzig.pdf