Nazareno Borello Marco Galatola a.a. 2019/2020

Relazione TLN - Parte 1

Traduttore interlingua EN -> IT

Panoramica

Lo scopo del progetto è quello di costruire un semplice traduttore Inglese -> Italiano in grado di gestire almeno le seguenti frasi:

- *you are imagining things*;
- *there is a price on my head*;
- *your big opportunity is flying out of here*.

La traduzione viene eseguita attraverso il passaggio da un'interlingua basata sulla logica del prim'ordine definita da una grammatica Context Free.

Per la generazione delle frasi italiane è stato utilizzato un Sentence Plan.

Implementazione

Il progetto è costituito da tre componenti principali:

- uno **script Python** per la parsificazione della frase e la gestione dell'I\O;
- una **Feature Context Free Grammar** per la definizione delle regole di traduzione da linguaggio naturale a forma logica;
- un'**applicazione Java** per la generazione delle frasi attraverso l'utilizzo di un sentence plan derivato dalla forma logica.

Script Python

Per parsificare le frasi date in input viene usato il parser incluso in NLTK al quale viene passata la grammatica.

```
parser = load_parser('res/grammar.fcfg')
```

La frase, suddivisa per parole, viene passata al parser che produce tutti i possibili alberi semantici relativi ad essa.

Da ogni albero viene estratta la semantica della radice, che consiste di una congiunzione di clausole in FOL, vengono considerate solo le logiche distinte.

```
tokens = sentence.split()
for tree in parser.parse(tokens):
    logic = tree.label()['SEM']
```

La logica viene passata al programma Java tramite il modulo di subprocess e ne viene registrato l'output in formato UTF-8.

```
output = subprocess.check_output(["java", "-jar", "res/NLG.jar", str(logic)
[1:-1]])
```

Sono gestiti i due possibili casi d'errore:

- il mancato riconoscimento di uno o più lessemi da parte del parser;
- l'impossibilità di costruire un albero semantico partendo dalla frase data in input.

Ognuno degli errori è segnalato tramite apposito messaggio e non causa un'interruzione del flusso di esecuzione.

Feature Context Free Grammar

La grammatica, scritta in formato fcfg, consiste di due tipi di regole di produzione:

- **regole grammaticali** per la scomposizione dei simboli non terminali;
- **regole lessicali** per la produzione e la traduzione in Italiano dei simboli terminali.

Di seguito le feature utilizzate:

- **SEM** - Semantica principale del nodo in FOL;
- **NUM** - Il numero del lessema (sg = singolare, pl = plurale);
- **PERS** - La persona di un pronome (1, 2 o 3);
- **TNS** - Il tempo di un verbo (pres = presente, past = passato);
- **MOD** - Il modo di un verbo (norm = normale, progr = gerundio);
- **LEX** - Il lessema associato ad un elemento, utilizzato come riferimento per le lambda espressioni;
- **TYPE** - Il tipo di un aggettivo (desc = qualitativo, poss = possessivo)
- **PROP** - Le sfumature del significato di un nodo non rappresentate nella semantica.

Le proprietà definiscono tutte le caratteristiche del nodo non includibili nella semantica poiché quest'ultima viene usata per il lambda calcolo e non può contenere più clausole. Non sono incluse fra le feature perché, non influenzando la costruzione delle regole, sono solo da riportare invariate all'interno della semantica della radice.
Le proprietà possibili sono:

- **pl(x)** - x è un nome ed è plurale, se questa proprietà non è presente allora x è singolare (pl(cosa) -> cose);
- **prep(x, y, a)** - a è una preposizione riferita ad x con argomento y (es. prep(fuori, qui, di) -> fuori di qui);
- **desc(x, a)** - a è un aggettivo descrittivo riferito ad x (desc(occasione, grande) -> grande occasione);

- **poss(x, a)** - a è un aggettivo possessivo riferito ad x (poss(testa, mio) -> mia testa);
- **adv(x)** - x è un pronome utilizzato come nome (es. *qui* in *fuori di qui*);
- **udf(x)** - x è un nome al quale è associato un articolo indeterminativo(es. udf(prezzo) -> un prezzo);
- **pn(x)** - x è un pronome.

Regole lessicali

Nome

```
N[NUM=<pl>,SEM=<testa>,PROP=<pl(testa)>] -> 'heads'
```

Pronome personale

```
PRP[NUM=<sg>,PERS=<1>,SEM=<io>] -> 'i'
```

Articolo indeterminativo

```
Det[NUM=<sg>] -> 'a'
```

Avverbio

```
RB[SEM=<\x.adv(x,fuori)>,LEX=<fuori>,PROP=<true>] -> 'out'
```

Aggettivo

```
JJ[SEM=<\x.desc(x,grande)>,TYPE=<desc>] -> 'big'
```

Preposizione

```
P[SEM=<\x.(\y.prep(y,x,di))>] -> 'of'
```

Verbo ausiliario

```
SV[NUM=<sg>,PERS=<3>,SEM=<\x.(\y.essere(y,x))>,TNS=<pres>,LEX=<essere>] -> 'is'
```

Verbo transitivo

```
TV[NUM=<sg>,PERS=<2>,SEM=<\x.(\y.immaginare(y,x))>,TNS=<pres>,MOD=<norm>,LEX=
<immaginare>] -> 'imagine'
```

Verbo intransitivo

```
IV[NUM=<sg>,PERS=<2>,SEM=<\x.volare(x)>,TNS=<pres>,MOD=<norm>,LEX=<volare>] ->
'fly'
```

Regole grammaticali

Start

```
S[SEM = <?vp(?subj) & tense(?l,?t,?m) & ?pr1 & ?pr2>] -> NP[NUM=?n,PERS=?p,SEM=?subj,PROP=?pr1] VP[NUM=?n,PERS=?p,SEM=?vp,TNS=?t,MOD=?m,PROP=?pr2,LEX=?l]
```

Caso particolare per prasi che iniziano con l'avverbio *ci*

```
S[SEM = <?sv(?obj,NONE) & tense(?l,?t,norm) & ?adv(?l) & ?pr>] -> RB[SEM=?adv,LEX=<ci>] SV[NUM=?n1,PERS=?p,SEM=?sv,TNS=?t,LEX=?l] NP[NUM=?n1,PERS=?p,SEM=?obj,PROP=?pr]
```

Frase Nominale

```
NP[NUM=?n,PERS=?p,SEM=?np,PROP=<pn(?np)>] -> PRP[NUM=?n,PERS=?p,SEM=?np]
NP[NUM=<pl>,PERS=<3>,SEM=?np,PROP=?pr] -> N[NUM=<pl>,SEM=?np,PROP=?pr]
NP[NUM=<sg>,PERS=<3>,SEM=?np,PROP=<udf(?np) & ?pr>] -> Det[NUM=?n] N[NUM=<sg>,SEM=?np,PROP=?pr]
NP[NUM=?n,PERS=?p,SEM=?np,PROP=<udf(?np) & ?pr>] -> Det[NUM=?n] NP[NUM=?n,PERS=?p,SEM=?np,PROP=?pr]
NP[NUM=?n,PERS=?p,SEM=?np,PROP=<?pr1 & ?pp(?np) & ?pr2>] -> NP[NUM=?n,PERS=?p,SEM=?np,PROP=?pr1] PP[SEM=?pp,PROP=?pr2]
NP[NUM=?n,PERS=?p,SEM=?np,PROP=<?pr & ?jj(?np)>] -> JJ[SEM=?jj] NP[NUM=?n,PERS=?p,SEM=?np,PROP=?pr]
NP[NUM=<sg>,PERS=<3>,SEM=?np,PROP=<?jj(?np) & ?pr>] -> JJ[SEM=?jj,TYPE=<poss>] N[NUM=<sg>,SEM=?np,PROP=?pr]
```

Caso particolare per un aggettivo qualificativo che viene applicato ad un nome al singolare

```
N[NUM=<sg>,PERS=<3>,SEM=?np,PROP=<?jj(?np) & ?pr>] -> JJ[SEM=?jj,TYPE=<desc>]
N[NUM=<sg>,SEM=?np,PROP=?pr]
```

Frase Verbale

```
VP[NUM=?n,PERS=?p,SEM=<?v(?obj)>,TNS=?t,MOD=<norm>,PROP=?pr,LEX=?l] -> TV[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=<norm>,LEX=?l] NP[SEM=?obj,PROP=?pr]
VP[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=<norm>,PROP=<true>,LEX=?l] -> IV[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=<norm>,LEX=?l]
VP[NUM=?n,PERS=?p,SEM=<?v(?obj)>,TNS=?t,MOD=<progr>,PROP=?pr,LEX=?l] -> SV[NUM=?n,PERS=?p,TNS=?t] TV[SEM=?v,MOD=<progr>,LEX=?l] NP[SEM=?obj,PROP=?pr]
VP[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=<progr>,PROP=<true>,LEX=?l] -> SV[NUM=?n,PERS=?p,TNS=?t] IV[SEM=?v,MOD=<progr>,LEX=?l]
VP[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=?m,PROP=<?pr1 & ?rb(?l) & ?pr2>,LEX=?l] ->
VP[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=?m,PROP=?pr1,LEX=?l] RB[SEM=?rb, PROP=?pr2]
VP[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=?m,PROP=<?pr1 & ?pp(?l) & ?pr2>,LEX=?l] ->
VP[NUM=?n,PERS=?p,SEM=?v,TNS=?t,MOD=?m,PROP=?pr1,LEX=?l] PP[SEM=?pp, PROP=?pr2]
```

Frase Preposizionale

```
PP[SEM=<?p(?l)>,PROP=<adv(?l) & ?pr>] -> P[SEM=?p] RB[LEX=?l,PROP=?pr]
PP[SEM=<?p(?np)>,PROP=?pr] -> P[SEM=?p] NP[SEM=?np,PROP=?pr]
```

Avverbio

```
RB[SEM=?rb,PROP=<?pr1 & ?pp(?l) & ?pr2>] -> RB[SEM=?rb,LEX=?l, PROP=?pr1]
PP[SEM=?pp,PROP=?pr2]
```

Applicazione Java

L'applicativo Java è composto da due classi:

- **NLG** (**N**atural **L**anguage **G**enerator) che si occupa dell'I\O;
- **LTS** (**L**ogic **T**o **S**entence) che si occupa di tradurre la logica in un Sentence Plan e di generare la frase.

Per generare le frasi è stata usata la libreria Java Simple NLG-it a sua volta derivata da Simple NLG.

NLG

La classe NLG consiste del solo metodo *main*, il quale prende in input un solo argomento (la frase in forma logica) e ne stampa la traduzione generata da LTS, opportunamente inizializzata col lessico italiano.

```
public static void main(String[] args) {
    LTS lts = new LTS(new ITXMLLexicon());

    System.out.println(lts.makeSentence(args[0]));
}
```

LTS

La classe LTS è invece più complessa ed è composta dal solo metodo pubblico *makeSentence* e da vari metodi privati.

La classe è caratterizzata da quattro attributi: un lessico, una *factory* per la costruzione dei sentence plan, un *realiser* per realizzarli ed una lista di clausole logiche.

```
private final Lexicon lex;
private final NLGFactory factory;
private final Realiser realiser;
private final ArrayList<String[]> clauses;
```

Il costruttore necessita soltanto del lessico, gli altri metodi sono discussi di seguito.

public String makeSentence(String logic)

Il metodo *makeSentence* traduce una frase FOL in linguaggio naturale

Come prima cosa la frase in FOL viene segmentata in una lista di vettori di stringhe, corrispondenti alle clausole, dal metodo *translate logic* che le salva nell'attributo *clauses*.

La grammatica genera sempre logiche aventi come prima clausola l'azione principale, contenente verbo, soggetto ed oggetto, per cui viene salvato il primo elemento di *clauses* nella variabile *core*.

```
translateLogic(logic);
String[] core = clauses.remove(0);
```

Successivamente ogni elemento del *core*, se esistente, viene espanso grazie agli appositi metodi *expand* e convertito in un *PhraseSpec*.

```
NPPhraseSpec subject = "NONE".equals(core[1]) ? null : expandNP(core[1]);
VPPhraseSpec action = expandVP(core[0]);
NPPhraseSpec object = core.length > 2 ? expandNP(core[2]) : null;
```

Siccome il verbo eredita il numero dall'elemento posto prima di esso in *createClause* e che alcune frasi iniziano con un predicato (es. *Ci sei tu*), è stato necessario introdurre un controllo che distingua questa composizione da quella base per poter impostare il numero programmaticamente.

```
SPhraseSpec sentence;

if (subject == null && object != null) {
    action.setPlural(object.isPlural());
    sentence = factory.createClause(action, object);
} else {
    sentence = factory.createClause(subject, action, object);
}
```

Il metodo ritorna la frase generata dal *realiser* sotto forma di stringa.

```
return realiser.realiseSentence(sentence);
```

private void translateLogic(String logic)

Il metodo *translateLogic* segmenta la logica in una lista di clausole.

Dal momento che la frase in forma logica consiste di una serie di clausole in AND, si può eseguire uno *split* sulle stesse, gli elementi di ogni clausola saranno poi organizzati all'interno di un vettore di stringhe.

Esempio di clausola:

```
"tense(immaginare,pres,progr)" -> ["tense", "immaginare", "pres", "progr"]
```

Il primo elemento del vettore equivale sempre al nome della proprietà mentre il secondo al lessema cui si fa riferimento.

private NPPhraseSpec expandNP(String lexeme)

L'insieme di metodi *expand_* trasforma ricorsivamente un lessema in un *phraseSpec*.

Il primo passo consiste nella creazione di una lista *refs* di riferimenti al lessema all'interno della logica, tramite il metodo *createRefs*.

```
ArrayList<String[]> refs = createRefs(lexeme);
```

Successivamente viene fatto uno *switch* su tutti i riferimenti, costruendo il sentence plan in base al tipo proprietà trovata.

```
for (String[] ref : refs) {
    switch (ref[0]) {
        case "desc" -> {
            WordElement we = lex.getWord(ref[2], LexicalCategory.ADJECTIVE);
            we.setFeature(ItalianLexicalFeature.QUALITATIVE, true);
            np.addModifier(we);
```

```
        }
        case "poss" -> {
            WordElement we = lex.getWord(ref[2], LexicalCategory.ADJECTIVE);
            we.setFeature(ItalianLexicalFeature.POSSESSIVE, true);
            np.addModifier(we);
        }
        case "adv" ->
            adverb = true;
            case "prep" -> {
            PPPhraseSpec pp = factory.createPrepositionPhrase(ref[3],
expandNP(ref[2]));
            np.addComplement(pp);
        }
        case "udf" ->
            article = "un";
        case "pl" ->
            plural = true;
        case "pn" ->
            pronoun = true;
    }
}
```

Il metodo *setPlural* della *factory* viene eseguito per ultimo poiché altrimenti non viene registrato correttamente.

L'articolo, di base determinativo, viene inserito solo se il lessema non è un pronome od un avverbio.

```
NPPhraseSpec np = factory.createNounPhrase(lexeme);

if (!(pronoun || adverb)) {
    np.setSpecifier(article);
    np.setPlural(plural);
}
```

Il phraseSpec viene infine ritornato.

Esistono metodi analoghi per la generazione di frasi verbali ed avverbiali.

Esempi

Frase inglese:

You are imagining things

Forma logica:

(immaginare(tu,cosa) & tense(immaginare,pres,progr) & pn(tu) & pl(cosa))

(immaginare(voi,cosa) & tense(immaginare,pres,progr) & pn(voi) & pl(cosa))

Frase italiana:

Tu stai immaginando le cose.

Voi state immaginando le cose.

Frase inglese:

There is a price on my head

Forma logica:

(essere(NONE,prezzo) & tense(essere,pres,norm) & adv(essere,ci) & udf(prezzo) & true & prep(prezzo,testa,su) & poss(testa,mio) & true)

Frase italiana:

Ci è un prezzo sulla mia testa.

Frase inglese:

Your big opportunity is flying out of here

Forma logica:

(volare(occasione) & tense(volare,pres,progr) & poss(occasione,tuo) & desc(occasione,grande) & true & true & adv(volare,fuori) & true & prep(volare,qui,di) & adv(qui) & true)

(volare(occasione) & tense(volare,pres,progr) & poss(occasione,vostro) & desc(occasione,grande) & true & true & adv(volare,fuori) & true & prep(volare,qui,di) & adv(qui) & true)

(volare(occasione) & tense(volare,pres,progr) & poss(occasione,tuo) & desc(occasione,grande) & true & true & adv(volare,fuori) & true & prep(fuori,qui,di) & adv(qui) & true)

(volare(occasione) & tense(volare,pres,progr) & poss(occasione,vostro) & desc(occasione,grande) & true & true & adv(volare,fuori) & true & prep(fuori,qui,di) & adv(qui) & true)

Frase italiana:

La tua grande occasione sta volando fuori di qui.

La vostra grande occasione sta volando fuori di qui.

Frase inglese:

She was imagining his prices

Forma logica:

(immaginare(ella,prezzo) & tense(immaginare,past,progr) & pn(ella) & pl(prezzo) & poss(prezzo,suo))

Frase italiana:

Ella stava immaginando i suoi prezzi.

Frase inglese:

There were their big heads

Forma logica:

(essere(NONE,testa) & tense(essere,past,norm) & adv(essere,ci) & pl(testa) & desc(testa,grande) & poss(testa,loro))

Frase italiana:

Ci erano le loro grandi teste.

Frase inglese:

I fly out

Forma logica:

(volare(io) & tense(volare,pres,norm) & pn(io) & true & adv(volare,fuori) & true)

Frase italiana:

Io volo fuori.

NOTA: il numero di traduzioni possibili equivale a 2^n dove n è il numero di ambiguità binarie