Laurea Specialistica in Informatica
a.a. 2007-2008

Interazione Uomo-Macchina II:

Laboratorio di Interfacce Intelligenti

Fiorella de Rosis

Esercitazione 4

Surface NLG

Tutor esercitazione: Nicole Novielli

Prerequisiti

Generazione di linguaggio: concetti di base su teorie e metodi

Materiale

- Dispense del corso, Unità 3 e 4 (reperibili sul sito web)
- Slide esercitazoni precedenti su formalizzazione (es. 1) e definizione di piani (es. 2)
- Slide dell'esercitazione (online dopo l'esercitazione in laboratorio)

Obiettivi

Esercitazione 2

Testo in linguaggio naturale
↓ *Analisi della struttura del Testo*
Struttura Intenzionale
Struttura Retorica
↓ *Definizione degli atti comunicativi*
Piano del Discorso

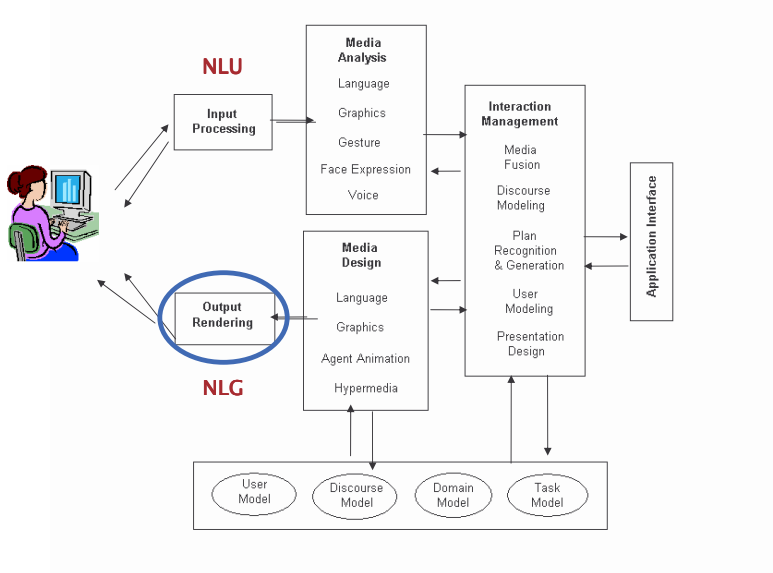
Esperienza con *RSTTool*

Esercitazione 4

Piano del Discorso
↓ *Trasformazione degli atti comunicativi in frasi*
Generazione superficiale

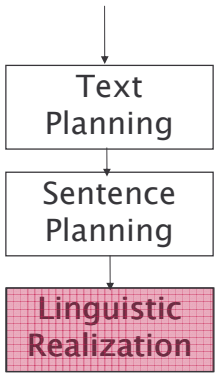
Esperienza con *SimpleNLG*

Architettura di una interfaccia intelligente (da Maybury e Wahlster)



Task e Architetture NLG

Architettura 'Pipeline'



Cosa non fa simplenlg:
(planning, scelta lessicale etc.)

Vediamo in dettaglio cosa fa simplenlg…

Cosa fa simplenlg

Ortografia:

- inserisce opportunamente gli spazi bianchi tra le frasi di un paragrafo
- gestisce in modo intelligente la punteggiatura ridondante (es. genera la frase "*He lives in Washington D. C.*" anziché "*He lives in Washington D. C..*"
- formatta automaticamente liste come "apples, pears and oranges" (punteggiatura e congiunzioni)
- gestisce l'organizzazione del testo sulle righe in modo da non frammentare le parole

Morfologia:

- gestisce l'accordo declinando i lessemi a seconda di genere, numero, tempo verbale (*tense*), persona

Simple grammar:

- assicura la correttezza grammaticale (es. accordo soggetto-verbo)
- genera correttamente le parti verbali (es. verbi con ausiliari come in "does not like")
- consente all'utente di definire la creazione di frasi tramite la combinazione di varie parti del discorso in strutture sintattiche appropriate

Come si utilizza

Simplenlg è un insieme di librerie per la generazione superficiale di frasi grammaticalmente corrette in Inglese.

Il package (con documentazione) si può scaricare al link http://www.csd.abdn.ac.uk/~ereiter/simplenlg/

Faremo riferimento alla versione 3.6 (la più recente)*

Si tratta di *librerie in Java*, sviluppate dal gruppo di NLG dell'università di Aberdeen, che possono essere integrate nel *vostro codice*.

* la versione 3.6 è corredata da tutorial dettagliato e documentazione dettagliata del codice

Esempio: come importare le librerie simplenlg in Eclipse

1. create a new Project (File>New>Project)
2. add the *simplenlg* library to the project's build path by doing the following:
· go to the Eclipse menu and select Project>Properties>Java Build Path
· click on the Libraries tab
· click on the Add External jars button
· browse to the *simplenlg* jar file which you downloaded from the web
· select the *simplenlg* jar file and click 'Open'
· click OK
3. In the project, create a new class which has the main method in it. In this example, let's call the new class TestMain.

4. At the top of the class, put in the following import statements

```
import simplenlg.features.*;
import simplenlg.lexicon.*;
import simplenlg.realiser.*;

public class TestMain {

    /**
     * @param args
     */
    public static void main(String[] args) {
          // TODO Auto-generated method stub

    }

}
```

Generazione di un paragrafo in simplenlg

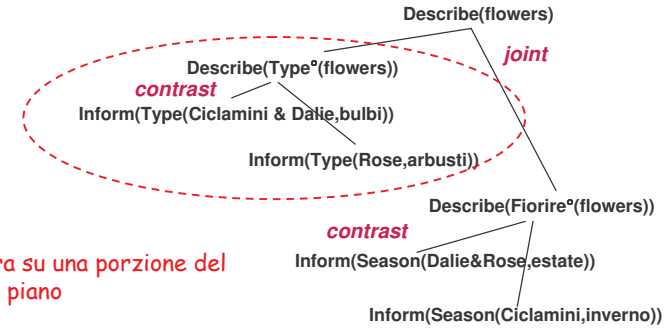
- Combinando frasi preconfezionate
- Generando dinamicamente le frasi da concatenare

In entrambi i casi la concatenazione di frasi va decisa in accordo col piano del discorso e le RR utilizzate (Unità 4 ed Esercitazione 2)

Vediamo come…

L'opzione più semplice: generazione di un paragrafo a partire da frammenti di testo 'preconfezionati'

Riprendiamo un esempio visto nell'unità 4

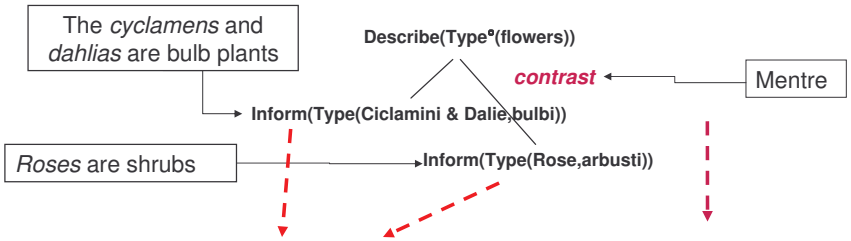


Concentriamoci per ora su una porzione del nostro piano

Supponiamo di avere un *database* in cui ad ogni formalizzazione logica di un atto comunicativo corrisponde una

frase preconfezionata…

L'opzione più semplice: generazione di un paragrafo a partire da frammenti di testo 'preconfezionati'



Ricerco nel db gli atti comunicativi associati alle foglie…

Inform(Type(Ciclamini & Dalie,bulbi))	The cyclamens and dahlias are bulb plants
Inform(Type(Rose,arbusti))	Roses ar shrubs
…	…

…e le congiunzioni associate alle RR

Joint	'And', ',', '.'
Contrast	'while'
…	…

Vedremo in seguito, nell'esercitazione sull'affective computing, come risolvere il problema dello stile: quale parola/costrutto/etc. è più idoneo, tra tante alternative, rispetto al mio obiettivo comunicativo (e non solo…)

Ho selezionato quindi i due frammenti di frase…

The *cyclamens* and *dahlias* are bulb plants

Roses are shrubs

… e voglio concatenarli usando il connettivo 'while'

Definisco due stringhe assegnando il valore estratto dal db

```
String str1 = "The cyclamens and dahlias are bulb plants";
String str2 = "roses are shrubs";
TextSpec t1 = new TextSpec();
t1.addSpec(str1);
t1.addSpec(str2);
t1.setListConjunct("while");
```

Creo una nuova istanza della classe che simplenlg utilizza per definire paragrafi.

Aggiungo a t1 i due frammenti…

E setto la congiunzione ('and' è la congiunzione di default)

```
Realiser r = new Realiser();
String output = r.realiseDocument(t1);
System.out.println(output);
```

Infine creo e invoco il realiser ottenendo:

The cyclamens and dahlias are bulb plants while roses are shrubs.

Simplenlg aggiunge automaticamente il punto alla fine della frase

Il Realiser

Classe che consente l'effettiva creazione del testo, ultimo passo del processo di generazione di una frase in linguaggio naturale

NOTA: è sufficiente crearne una sola istanza volta, all'inizio.

Una volta creata un'istanza *r* di Realiser…

```
Realiser r = new Realiser();
String output = r.realiseDocument(t1);
System.out.println(output);
```

Posso invocarla *n* volte per generare *n* diverse frasi o paragrafi in linguaggio naturale

```
String str1 = "The cyclamens are bulb plants";
String str2 = "roses are shrubs";

TextSpec t1 = new TextSpec();
t1.addSpec(str1);
t1.addSpec(str2);
String output = r.realiseDocument(t1);
System.out.println(output);
```

→ *"The cyclamens are bulb plants while roses are shrubs"*

```
TextSpec t2 = new TextSpec();
String str3 = "My dog likes bones";
t2.addSpec(t3);
output = r.realiseDocument(t2);
System.out.println(output);
```

→ *"My dog likes bones"*

Simplenlg gestisce le joint multiple adeguando la punteggiatura

```
String str1 = "My cats like fish";
String str2 = "my dog likes bones";
String str3 = "my horse likes grass";
TextSpec t1 = new TextSpec(); // create a TextSpec
t1.addSpec(str1);
t1.addSpec(str2);
t1.addSpec(str3);
```

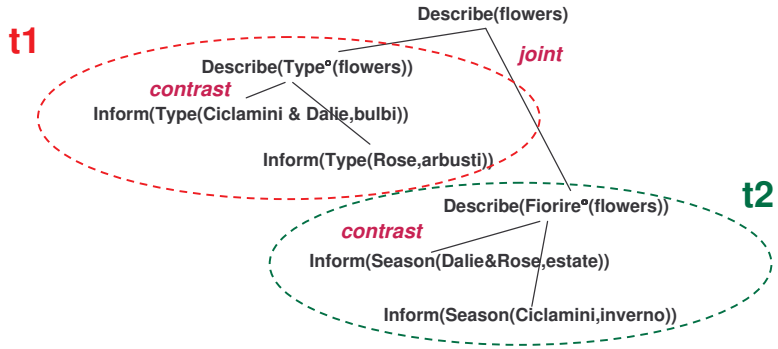
Produce in output il paragrafo:

*"My cats like fish, my dog likes bones **and** my horse likes grass"*

Poichè 'and' è la congiunzione di default e non ne viene specificata alcuna

Torniamo al nostro piano…

Con la stessa tecnica illustrata per la prima porzione, genero la seconda parte di piano



A questo punto ho le due porzioni di piano, t1 e t2:

t1 = " *The cyclamens and dahlias are bulb plants while roses are shrubs.*"

t2 = "*Dahlias and roses bloom in summer while cyclamens flourish in winter.*"

Che voglio combinare tramite una relazione retorica di **'Joint'**


```
Realiser r = new Realiser();

TextSpec t1 = new TextSpec();
String s1 = new String("The cyclamens and dahlias are bulb plants");
String s2 = new String("roses are shrubs");
t1.addSpec(s1);
t1.addSpec(s2);
t1.setListConjunct("while");

TextSpec t2 = new TextSpec();
String s3 = new String("Dahlias and roses bloom in summer");
String s4 = new String("cyclamens flourish in winter");
t2.addSpec(s3);
t2.addSpec(s4);
t2.setListConjunct("while");

TextSpec t3 = new TextSpec();
t3.addSpec(t1);
t3.addSpec(t2);

String output = r.realiseDocument(t3);
System.out.println(output);
```

The cyclamens and dahlias are bulb plants while roses are shrubs

Dahlias and roses bloom in summer while cyclamens flourish in winter

Nota: nel concatenare due istanze di TextSpec (paragrafi), simplenlg usa automaticamente il punto, come congiunzione

The cyclamens and dahlias are bulb plants while roses are shrubs. Dahlias and roses bloom in summer while cyclamens flourish in winter

Abbiamo visto un esempio semplice di generazione di un paragrafo:

1. ***seleziono*** da un db le ***frasi preconfezionate*** associabili agli atti comunicativi riportati nei nodi foglia del nostro piano
2. ***seleziono*** dal db le ***congiunzioni*** corrispondenti alle relazioni retoriche che legano gli atti comunicativi
3. ***concateno*** utilizzando i metodi della classe ***TextSpec***, usata da simplenlg per rappresentare i paragrafi

MA

Necessità di personalizzare (unità 4) le frasi del mio messaggio rispetto (per esempio) al mio User Model:

- quale ***stile*** usare (colloquiale, formale, …)
- quale ***lessico*** usare
- in che ***posizione relativa*** mettere i diversi elementi (ottenendo diversi effetti comunicativi)

Ecc…

↓

Creazione dinamica delle frasi del mio paragrafo

Come genero una frase semplice

1. Creo un'istanza della classe **SPhraseSpec** (che rappresenta la nostra frase)
2. creo un'**istanza** per ognuna delle varie 'Part Of Speech' (**POS**) che la compongono
3. specifico il verbo
4. **combino le POS** specificandone il ruolo all'interno della frase (soggetto, complemento)
5. setto il TENSE adeguato scegliendo tra presente (default), passato e futuro
6. specifico la forma: dichiarativa (default) o interrogativa
7. nel caso di **subordinate**, specifico la congiunzione che esprime il rapporto con il nucleo

(vedremo con degli esempi in quali modi posso chiedere a simplenlg di rendere in linguaggio naturale le relazioni retoriche che legano le frasi del nostro piano)

Un esempio molto semplice

The Cyclamens and Dahlias are bulb plants

soggetto — verbo — Complemento oggetto (diretto)

Una volta individuati i ruoli delle varie POS…

Cyclamens and dahlias (Noun Phrases) = soggetto
Bulb plants (Noun Phrase) = complemento oggetto
are (verb) = il verbo della mia frase

…setto opportunamente gli elementi di un'istanza di SPhraseSpec

```
SPhraseSpec p = new SPhraseSpec();

NPPhraseSpec subj1 = new NPPhraseSpec("Cyclamens");
subj1.setDeterminer("The");

NPPhraseSpec subj2 = new NPPhraseSpec("Dahlias");

NPPhraseSpec compl = new NPPhraseSpec("bulb plants");

p.setSubject(subj1);
p.addSubject(subj2);

p.setVerb("be");
p.addComplement(compl);
```

Creo le noun phrase per soggetto e complemento oggetto utilizzando la classe NPPhraseSpec

Simplenlg accorda il verbo con il soggetto (III persona plurale)

Mapping tra formalizzazione e input per simplenlg

Di default, simplenlg genera frasi al modo indicativo, tempo presente simulando, di fatto, delle *Inform*

L'atto comunicativo ci dà informazioni sulla forma (dichiarativa, in questo caso) con cui rendere la frase

Il predicato può essere utilizzato per decidere quale verbo utilizzare (to be)

Inform (IsA (Ciclamini & Dalie,bulb plants))

Il primo argomento del predicato potrebbe diventare il soggetto…

Mentre *il secondo* potrebbe diventare il complemento oggetto

Agendo sulle feature dell'istanza *p* di SPhraseSpec posso rendere diversi *atti comunicativi*

AskIf

p.setInterrogative(InterrogativeType.YES_NO);

Are the Cyclamens and Dahlias bulb plants.

AskAbout

p.setInterrogative(InterrogativeType.WHAT, DiscourseFunction.OBJECT);

What are the Cyclamens and Dahlias?

p.setInterrogative(InterrogativeType.WHAT,DiscourseFunction.SUBJECT);

What are bulb plants

AskJustify

p.setInterrogative(InterrogativeType.WHY);

Why are the Cyclamens and Dahlias bulb plants?

Simplenlg mi consente di simulare tutti i tipi di Wh question, oltre a domande di tipo Yes/No, How ecc…

Un caso più complesso

Suggest (ShouldDo(U,Running))

abbiamo bisogno di un verbo modale

"you" (ipotizziamo lo scenario di un sistema di dialogo)

Complemento oggetto

```
SPhraseSpec run = new SPhraseSpec("you", "go", "running");

run.setModal("should");
```

"You should go running"

Ma il sistema potrebbe voler rendere in modo leggermente diverso il medesimo atto comunicativo

Suggest (ShouldDo(U,Running))

```
SPhraseSpec run = new SPhraseSpec("you", "go", "running");
run.setModal("should");
```

Aggiunta di **'ingredienti' linguistici**

che danno una connotazione lievemente differente al mio messaggio

```
run.addModifier("really");

SPhraseSpec think = new PhraseSpec("I", "think");
think.addComplement(run);

run.suppressComplementiser(true);
```

Elimina il "that" restituendo in output *"I think you should really go running"*

Ne parleremo più in dettaglio nell'esercitazione sull'affective computing...

L'esempio sulla persuasione

"You should go running, Giuseppe! You are young, you said you want to be in a good shape and running helps you to mantain a good shape"

ShouldDo((RUNNING)
Likes(G, GoodShape) *and*
Implies(Running,GoodShape).
In addition
CanDo(G,RUNNING) *as*
(Healthy(G) *and* Young(G))

La scelta dell'atto comunicativo si riflette sulla struttura del messaggio in output!

Abbiamo visto il caso della Suggest

Suggest (ShouldDo(U,Running)) → **Verbo modale** → *You should go running, Giuseppe!*
I think you should go running, Giuseppe!

PROBLEMA: come fareste (formalizzazione e conseguente sentence planning) a rendere varianti come:

Go running, Giuseppe! — Quale atto comunicativo?

You are young — Quale modo e tempo?

Running helps you to mantain a good shape

Etc...

Nuclei e subordinate

Poichè non è un sentence planner, simplenlg non gestisce esplicitamente ruolo delle frasi rispetto alla RR che la lega

Per realizzare la coordinazione (più nuclei in joint) e la subordinazione (nucleo + satellite) posso:

a. **Soluzione banale**: specificare qualsiasi congiunzione tra due TextSpec (e in quel caso otterò la coordinazione tra due nuclei o la subordinazione tra nucleo e satellite semplicemente tramite il setlistconjuntion

b. **Soluzione più raffinata** (concettualmente più idonea a rappresentare il legame tra una subordinata e il nucleo): Specificare due SSPhrase e aggiungere la 'cue' prase alla seconda, la subordinata

PROBLEMA: come realizzereste frasi come:

You should go running

because

you are young

Le scelte implementative spettano a chi realizza il codice di Sentence Planning (la seconda opzione è più lungimirante ed assicura maggiore flessibilità nella generazione dinamica di frasi)

I complementi indiretti

Nell'esempio "*The Cyclamens and Dahlias are bulb plants*", "bulb plants" è il complemento diretto della frase

Problema: come gestisco i complementi indiretti?

Es: The flight arrives **to Denver**, **at 22 pm**

Per simplenlg un complemento è qualsiasi cosa venga DOPO IL VERBO

1. The dog chases the cat. — *Noun Phrase*
2. The dog is happy. — *Adjective*
3. The dog ran quickly. — *Adverb*
4. The dog chases the cat in the park. — *Prepositional Phrase*

Indipendentemente dalla parte del discorso (POS)!!!

Gli unici POS che simplenlg gestisce esplicitamente sono:

- le Noun Phrase ("*the cat*") ---> utilizzando la classe NPPhraseSpec

(ed abbiamo visto come gestirli, ad es. per il soggetto di una frase)

- le Prepositional Phrase (es. "*to Denver*") --- > utilizzando la classe PPPhraseSpec

Vediamo come…

*The flight arrives **to Denver**, **at 22 pm***

Effettuiamo il mapping con la stessa logica illustrata sino ad ora

Indicativo presente — *Soggetto* — *1° Complemento indiretto*

Inform (Arrive("The flight","","Denver","22 pm"))

Verbo — *Nessun compl. diretto (verbo intransitivo)* — *2° Complemento indiretto*

```
SPhraseSpec p = new SPhraseSpec();

NPPhraseSpec subj = new NPPhraseSpec("fligth");
subj.setDeterminer("The");
p.setSubject(subj);
NPPhraseSpec where = new NPPhraseSpec("Denver");

String prep = new String();

prep = "to";

PPPhraseSpec pp = new PPPhraseSpec(prep, where);
p.addModifier(pp);
```

etc… (analogamente per l'ora)

Il complemento indiretto ha bisogno di una preposizione…

… e va aggiunto come 'Modifier' alla mia frase

I complementi indiretti: problemi

Inform (Arrive("The flight","","Denver","22 pm"))

Problema: Come decido quale preposizione usare?

Simplenlg non supporta questo tipo di funzionalità:

non è un pianificatore nè 'comprende' o consente di gestire esplicitamente il tipo di complemento che voglio realizzare!!!

Possibile soluzione: sviluppo di un modulo **ad hoc** che

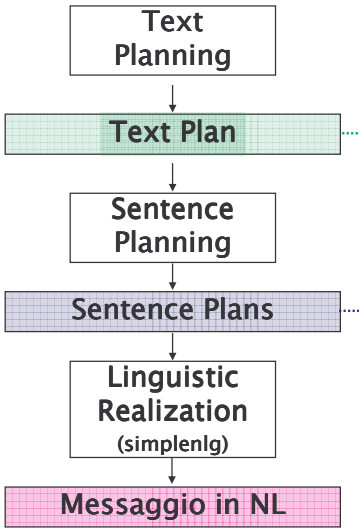
1. **riconosca** 'Denver' come nome proprio di città (ad esempio, facendo riferimento a dizionari semantici come WordNet)
2. **selezioni** il verbo opportuno, in base al nome del predicato (in questo caso 'to arrive')
3. **inferisca** che, sulla base di (1) e (2), voglio realizzare un complemento di moto a luogo, cui corrisponde (in questo caso) la proposizione 'to'

(NOTA: non sarebbe lo stesso se volessi dire Going(John, home)!!!

idem per la generazione del complemento di tempo 'at 22' (in teoria potrei avere una lista di complementi di cardinalità *n*, e dovrei poter ripetere il mapping per ognuno di essi)

Riepilogo

Simplelng agisce **SOLO** al livello della 'Linguistic Realization'



Partendo dal DPML…

```
<DPML>
 <goal name="Describe" term="Weather(x)"  RR="Contrast">
   <goal name="Describe" term="Weather(HZ1)"  RR="Joint">
     <communicative_act name="InformAbout"
term="List°(HZ1)" type="N"> </communicative_act>
                                                    [...]
<goal name="Describe" term="Weather(x2)" RR="Joint"> …
…</goal></DPML>
```

…è necessario sviluppare codice 'collante' che faccia da sentence planner…

Input per Simplenlg

Ho bisogno di definire:
- Lessico *(es. dizionari semantici come WordNet)*
- Rapporto tra le varie porzioni di frase (nuclei e subordinate) *(es. tramite table look-up)*
- Ruolo delle varie POS all'interno di ciascuna frase *(definendo procedure ad hoc in accordo con la formalizzazione utilizzata)*

ecc…

Problemi aperti: l'anafora

Riprendiamo l'esempio della postura e proviamo a generare il messaggio semplicemente concatenando gli atti comunicativi
Otterrei un testo di questo tipo

Ripetizioni

Inform S U Cause(BadPosture, Accidents);
A bad posture is the main cause of accidents or problems

Inform S U CauseInParticular(BadPosture, SeriousProblems, AgedPerson);
*Very often **a bad posture** is the cause of serious probles, especially for aged persons*

Inform S U Favour(CorrectExercise, BetterPosture);
Doing exercises in the right way favours a better posture,

Inform S U Favour(CorrectPosture,StrongerMuscles);
A correct posture** strengthens **the muscles

Inform S U Favour(CorrectPosture,LongerMuscles);
A correct posture** lengthens **the muscles

Inform S U Favour(LongerMuscles,RightPosition)
*Longer **muscles** help you in mantaining the right position*

Necessità di introdurre pronomi

Possibili soluzioni

Necessità di ottimizzare il testo

A bad posture is the main cause of accidents or problems
*Very often **a bad posture** is the cause of serious probles, especially for aged persons*
Doing exercises in the right way favours a better posture,
A correct posture** strengthens **the muscles
A correct posture** lengthens **the muscles
*Longer **muscles** help you in mantaining the right position*

Gestendo, ad esempio, l'anafora con l'uso di pronomi e l'aggiunta di avverbi

A bad posture is the main cause of accidents or problems
*Very often **it** is **also** the cause of serious probles, especially for aged persons*

riorganizzando l'intero periodo

Doing exercises in the right way favours a better posture,
***A correct posture** strengthens*
and** lengthens **the muscles

introducendo delle congiunzioni

***which** help you in mantaining the right position*

Come lo implemento a livello si sentence planner?

Problemi aperti: la scelta del lessico

Simplenlg non è in grado di selezionare la parola da utilizzare a partire da un concetto poiché non gestisce esplicitamente ontologie, dizionari semantici e non, né alcuna risorsa di questo tipo.

La scelta del lessico spetta interamente a chi implementa la componente di Sentence Planning

La scelta del lessema da impiegare può essere guidata da *varie ragioni*

Problema che affronteremo meglio nell'esercitazione sull'affective computing

Per l'esercitazione…

Concentratevi su uno o più tra i problemi relativi alla generazione superficiale (*passaggio da formalizzazione logica ad input per simplenlg con particolare attenzione a combinazione di nucleo e subordinata, gestione delle ripetizioni, scelta dell'atto comunicativo e influenza sullo stile del messaggio finale etc.*) prendendo spunto da quanto visto a lezione, e provate a sviluppare delle soluzioni