CURSO DE PROCESAMIENTO DE LENGUAJE NATURAL
2020/2021

SEGUNDO EJERCICIO PRÁCTICO

Descripción del problema

Se quiere implementar un reconocedor de voz que interprete órdenes de un aparato de cocina. El aparato puede realizar sólo dos operaciones, batir y calentar, que puede realizarse a diferentes velocidades, temperatura y durante un cierto tiempo. Además, el aparato responde a órdenes para detenerse o reanudar. La velocidad máxima y mínima es 10 y 1. La temperatura máxima y mínima es 100 y 0. El tiempo solo cubre minutos y segundos. Además, hay 10 programas predefinidos. Suponemos que hay un proceso previo para el tratamiento de voz, y se parte de una transcripción textual de las órdenes. Las órdenes se procesan secuencialmente. El trabajo a realizar en esta práctica consiste en abstraer la semántica de la orden a partir de la frase en lenguaje natural. Para ello, se definirá un modelo semántico mediante una estructura de rasgos y se aplicarán técnicas de análisis basadas en unificación (sección 18.4 del libro base), cubriendo al menos las siguientes frases:

“calienta a cincuenta grados dos minutos”
“bate a velocidad cinco noventa grados durante tres minutos y veinte segundos”
“bate a temperatura noventa diez minutos y medio”
“calienta a noventa grados durante dos minutos”
“batir dos minutos a temperatura veinte”
“revolver a temperatura máxima”
“girar a velocidad mínima”
“tres minutos a cien de temperatura y tres de velocidad”
“calentar a cincuenta grados a velocidad tres”
“para”
“detente”
“anular”
“cancelar”
“reanuda”
“continúa”
“continuar”
“activa el programa tres”
“inicia el programa cuatro”

Además, para evitar accidentes, el sistema no debe aceptar expresiones ambiguas o incorrectas como:
“batir a velocidad diez grados”
“girar a velocidad cien”
“activa el programa cien”
“detente a velocidad diez”

En caso de que la frase omita algún aspecto, (por ejemplo, temperatura en "batir a velocidad 3") el sistema automáticamente tomará el valor cero. No es necesario indicar los rasgos omitidos en la representación semántica. Como variante opcional, se puede ampliar el sistema para que acepte frases compuestas del tipo "bate durante 5 minutos y después calienta otros 5 minutos a 37 grados".

Algunas consideraciones a tener en cuenta son:

1. Por sencillez, asumiremos números del uno al diez, y múltiplos de diez. Dadas las limitaciones del tratamiento léxico en la herramienta empleada en esta práctica, es necesario introducir cada número con una entrada independiente. Por supuesto esto no sería así en un entorno real en donde un análisis léxico previo reconocería estos elementos como valores.
2. Es necesario tener en cuenta sinónimos (en nuestro contexto), como "reanudar" y "continuar".
3. Es necesario en algunos casos tener en cuenta el contexto de las palabras. Por ejemplo, un número puede referirse a la temperatura o al tiempo dependiendo de si viene precedido de ciertas palabras.
4. En la medida de lo posible se ha de tener en cuenta variaciones de estas frases.

Cuatro días antes del plazo de entrega pondremos disponible en el curso virtual un lote de frases de test. Estas frases de test no incluirán palabras que no aparezcan en las anteriores, aunque puede haber otro tipo de variaciones lingüísticas, como cambios en el orden de las palabras o en los valores numéricos. El objetivo de este test no es en cualquier caso conseguir un acierto total en un texto nuevo, sino tener la oportunidad de analizar la potencia de la aproximación y la dificultad de abordar la variabilidad lingüística.

Paso 1: Definición de reglas sintácticas

En el capítulo 12 del libro base "Speech and Language Processing", se describen las nociones básicas de la sintaxis y gramáticas, en concreto constituyente, relaciones gramaticales, subcategorización y dependencias. En particular, el primer paso para el desarrollo de esta práctica es la definición de una **gramática de contexto libre** que cubra las frases anteriores. Esto incluye las reglas de producción y el lexicón (sección 12.2).

Desde un punto de vista formal, los **símbolos terminales** se introducen en el lexicón. En el caso de nuestro dominio, éstos son el vocabulario que aparece en las frases descritas. Se valorará el añadir vocabulario adicional, como otras formas verbales, sinónimos de vehículo, etc. Por simplicidad consideraremos sólo 10 números de calle o plaza. En nuestro dominio, sólo hay tres tipo de vía, que pueden ser calle, plaza y avenida.

Por otro lado, los **símbolos no terminales** representan una generalización de los símbolos terminales. Por ejemplo, el símbolo no terminal asociado a cada símbolo terminal es su categoría léxica. En nuestro contexto consideraremos verbos, preposiciones, nombres, números, adjetivos y determinantes. En un nivel superior consideraremos sintagmas nominales, preposicionales y verbales.

La primera tarea a realizar en esta práctica consiste en definir nuestra gramática de contexto libre. Esta debe quedar definida en la memoria de la práctica, siguiendo la notación estandar (no implementar en Python). Tomar como ejemplo la figura 13.1 de capítulo 13 del libro base. La gramática debe reconocer **al menos** las frases propuestas al comienzo de este enunciado. Esto quiere decir que la gramática debería de poder reconocer otras frases. El sistema debe descartar frases ambiguas como que no deberían generar información semántica.

Paso 2: Representación semántica

El segundo paso de esta práctica consiste en definir una estructura de rasgos que cubra la semántica de las ordenes. En el capítulo 17 del libro base, en la sección 17.1 se describe algunas características deseables para un sistema de representación semántica. El primero de ellos es la **verificabilidad.** Esto quiere decir que el sistema debe conectar la información textual con información estructurada en una base de datos. Idealmente, la información extraída del texto debería ser **no ambigua**, aunque, como se describe en el libro, es necesario tener en cuenta la **vaguedad.** Por ejemplo, en nuestro contexto, un usuario puede indicar la temperatura pero no la velocidad. En ese caso, el sistema debería capturar la información aportada en el texto, independientemente de que sea o no completa. El sistema preguntará al usuario la información que falte. Los requisitos de inferencia (sección 17.1.4) y expresividad (sección 17.1.5) no son de relevancia en este ejercicio.

A continuación debemos de definir un **modelo** semántico para este problema (sección 17.2 del libro base). Un modelo es un que representa el estado particular de la realidad que estamos tratando de representar. Tres nociones básicas en el desarrollo de un modelo son los **objetos** (elementos del dominio), **propiedades** (conjuntos del dominio) y **relaciones** (conjuntos de tuplas de elementos del dominio). Estos tres tipos de elementos configuran el **vocabulario no lógico** de nuestro modelo. En nuestro caso, por ejemplo, queremos modelar una orden, es decir, detenerse o activarse a una cierta velocidad y temperatura durante un periodo de tiempo. Nuestro dominio estará formado por tanto el tipo de orden y sus características.

En esta práctica vamos a optar por modelar nuestro dominio mediante sucesos desde el enfoque **Davidsoniano** (sección 17.4 del libro base). En este enfoque, se declara una variable de suceso *e* que en nuestro caso representara el servicio. Básicamente, tendremos que identificar predicados sobre un determinado servicio (suceso *e*) del tipo: *acción(e, batir), minutos(e,2-minutos, 3 segundos)* o *velocidad(e, 6)*.

En esta práctica, representaremos la información semántica aumentando a nivel semántico las estructuras de rasgos sintácticas, empleando para ello **estructuras de rasgos** y **unificación** (ver capítulo 18 del libro base). En el libro base se describen técnicas para extraer proposiciones lógicas con variables y cuantificadores mediante estructuras de rasgos. Sin embargo, como se indica en la página 338 del libro *"Natural Language Processing with Python"*, que empleamos ya en la práctica anterior, las estructuras de rasgos son una herramienta de propósito general que permite representar conocimiento. En esta práctica, por simplicidad, obviaremos los cuantificadores y proposiciones lógicas y representaremos directamente el modelo mediante las estructuras de rasgos. En dicho libro aparecen ejemplos en los que se modela el concepto de persona, nombre, edad, dirección etc. En esa misma línea, en esta etapa debe definirse en la memoria de la práctica una estructura de rasgos (sin necesidad de implementarla en Python) que se corresponda con el conocimiento que queremos representar en nuestro dominio.

Paso 3: Familiarización con el entorno

El primer paso consiste en la familiarización con el lenguaje Python y su uso en la construcción de gramáticas basadas en rasgos. En los capítulos 8 y 9 del libro *"Natural Language Processing with Python"*, al que se puede acceder mediante los enlaces *http://www.nltk.org/book/ch08.html y http://www.nltk.org/book/ch09.html* aparece información sobre cómo implementar sobre NLK rasgos y restricciones, estructuras de rasgos y procesos de unificación mediante Python.

Paso 4:Definicion de la gramática extendida:

El siguiente paso en esta práctica consiste en la definición de la gramática de forma que se generen los atributos y valores necesarios para alimentar la base de datos. Para ello, tomaremos como ejemplo los ficheros *programa_base.py* y *gramatica_base.fcfg* disponibles en el material del curso virtual. La gramática base contiene el siguiente código:

```
% start S
# ###################
# Grammar Productions
# ###################
# S expansion productions
S[SEM=?sem] -> VP[SEM=?sem]
VP[SEM=[ACCION=?val]] -> V[ACCION=?val]


# ###################
# Lexical Productions
# ###################
V[ACCION=anular] -> 'anular' | 'cancelar'
```

En esta gramática, la estructura de rasgos a extraer se almacena en *SEM*. Es decir, toda la información semántica relativa nuestro modelo debe de aparecer en la estructura SEM en la respuesta del sistema. En este caso, la estructura resultante consta del rasgo *Acción*. Por supuesto, la estructura definitiva debería de ser más compleja.

La gramática debe extraer la semántica de las palabras según su contexto empleando la herramienta de unificación. Para entender el proceso vamos a tomar como referencia el siguiente ejemplo. En la siguiente regla asociada a un grupo nominal, se identifica un número como velocidad.

```
NP[SEM=[VELOCIDAD=?val]] -> Det_num[NUMERO=?val] N[SEMCAT=velocidad]
```

Donde en el lexicón tendremos:

```
Det_num[NUMERO=1] -> ‘uno’ | ‘dos’
N[SEMCAT=velocidad] -> ‘velocidad’
```

La estructura final generada se generará en la estructura SEM. El rasgo SEMCAT permite generalizar palabras con la misma categoría semántica como temperatura y grados. Es importante tener en cuenta que la técnica de unificación permite unificar en una única estructura diferentes estructuras generadas en diferentes ramas del árbol. Por ejemplo, la siguiente regla:

S[SEM=?s] -> VP[SEM=?s] NP[SEM=?s]

siempre que la estructuras semánticas extraídas en `VP` y `NP` sean compatibles, genera una única estructura en `S` conteniendo toda la información. Esta unificación tendrá en cuenta por tanto las restricciones expresadas mediante subcategorización en las reglas de NP y VP.

Además, el sistema debe no generar información semántica cuando la frase sea incoherente (para evitar desastres en la cocina). Por ejemplo, ante la frase “batir a velocidad diez grados”, para que el mecanismo de unificación no proceda, el símbolo no termina asociado a “diez” debe tener un rasgo semántico que le asocie a temperatura o velocidad, de forma que no pueda haber

unificación cuando la gramática le asigna dos categorías a la vez. Por ejemplo, para descartar frases como “activa el programa cien”, el símbolo no terminal asociado a “cien” debe tener una categoría de escala (1 o 10) o (10 a 100), que en este caso no encaja con la categoría semántica de “programa”. Para descartar órdenes como “detente a velocidad diez”, el símbolo no terminal “velocidad” debe estar asociado a la categoría semántica de “batir”, que no encaja con la categoría de “detente”. Debe por tanto especificarse estas restricciones por medio de técnicas de unificación.

Por otro lado, el programa base contiene el código:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
reload(sys)
sys.setdefaultencoding('utf8')

import nltk
from nltk import load_parser

cp = load_parser('gramatica_base.fcfg', trace=2)


infile = open('textos.txt')
# Mostramos por pantalla lo que leemos desde el fichero
for line in infile:
    print(line)
    tokens=line.split()
    trees = cp.parse(tokens)
    for tree in trees: print(tree)
infile.close()
```

El resultado de ejecutar este código (ejecutando en línea de comandos `python programa_base.py`) debería de contener:

```
cancelar
…
(S[SEM=[ACCION='anular']]
  (VP[SEM=[ACCION='anular']] (V[ACCION='anular'] cancelar)))
…
```

La práctica consiste en extender/modificar la gramática de forma que todas las frases del archivo *textos.txt* sean procesadas generando una estructura semántica de estas características pero incluyendo toda la información semántica. Se valorará el reconocer otras variantes de estas mismas frases. Es importante tener en cuenta que no es necesario que el sistema

MODO DE ENTREGA: Se entregará entregarse un fichero comprimido en zip conteniendo la memoria del trabajo con:

1. **La definición de la gramática sintáctica.**
2. **La definición de la estructura de rasgos que representa el modelo semántico.**
3. **La gramática extendida en formato NLTK.**
4. **El lexicón desarrollado también en formato NLTK.**
5. **Un apartado de evaluación en base a los resultados obtenidos sobre el fichero de test proporcionado y conclusiones acerca del potencial y limitaciones de estas técnicas a este problema.**

Además, el archivo comprimido debe incluir: la gramática, el programa base en caso de haberse realizado alguna modificación y un fichero con la salida.