CURSO DE PROCESAMIENTO DE LENGUAJE NATURAL
2019/2020

PRIMER EJERCICIO PRÁCTICO: IMPLEMENTACIÓN DE UN ANALIZADOR MORFOLÓGICO BÁSICO MEDIANTE EXPRESIONES REGULARES EN PYTHON.

En este ejercicio práctico se aplicarán expresiones regulares (capítulo 2 del libro base) en el análisis morfológico. Para ello se empleará el lenguaje Python y librerías de NLTK. El objetivo es definir un diccionario para palabras comunes y una serie de expresiones regulares que reconozcan las categorías gramaticales de las palabras contenidas en un texto.

Objetivos

En este ejercicio procesaremos el texto contenido en el fichero "texto.txt", que contiene descripciones de conceptos de mecánica de coches como *filtro, aceite*, *bomba*, *colector*, etc. Nos centraremos por tanto en un dominio de lenguaje concreto. Por lo general, el procesamiento de lenguaje natural se enfrenta a múltiples problemas derivados de la versatilidad del lenguaje: un vocabulario amplio, términos ambiguos, multilingüismo, etc. Por ello, el diseño específico de herramientas orientadas a un dominio determinado ofrece mejores resultados que soluciones genéricas para cualquier tipo de texto.

En esta práctica supondremos que queremos implementar un sistema automático de que atienda consultas (buscador de respuestas) en el dominio de coches. Para ello necesitaremos procesar sintáctica y semánticamente los textos, para lo que es necesario previamente un análisis morfológico. El objetivo es desarrollar y evaluar un analizador morfológico sobre textos dentro del dominio de mecánica de coches..

En nuestra aproximación, combinaremos el uso de diccionarios con expresiones regulares. La primera aproximación consiste en dotar al sistema de una lista de palabras con su información morfológica. Esta aproximación es efectiva sobre todo para términos muy frecuentes en el dominio o también para términos muy comunes como conjunciones, signos de puntuación o determinantes. Por otro lado, la morfología de términos como "lubrica**ción**", "circula**ción**", "admis**ión**", etc, puede derivarse mediante expresiones regulares a partir de su terminación en "**ión**", al menos dentro del dominio en el que nos encontramos. Para capturar estos patrones se emplean expresiones regulares.

En el contexto del procesamiento de lenguaje ninguna aproximación al problema es perfecta. El objetivo de esta práctica es experimentar estas limitaciones, y desarrollar una solución buscando un equilibrio entre:

- **Efectividad**. El sistema debería de ser capaz de procesar correctamente el texto sobre el que estamos trabajando. Por ejemplo, un diccionario con todos los términos posibles sería altamente efectivo.

- **Generalidad**. El sistema debería de procesar correctamente otros textos del dominio. En este sentido, un diccionario con todos los términos vistos hasta el momento no tendría generalidad. Para conseguirla se pueden usar expresiones regulares que modelan patrones genéricos, aunque a costa de la efectividad.

Instalación de Python y NLTK

Se encuentra disponible en internet el libro “Natural Language Processing with Python”. Este documento será la guía fundamental para la realización de las prácticas de este curso. Contiene las directrices para la instalación de herramientas. En cualquier caso, resumimos los pasos básicos en este documento.

En primer lugar, desde el enlace http://python.org/ se puede descargar Python. Se trata de un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Es un lenguaje de programación multi-paradigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,1 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores (https://es.wikipedia.org/wiki/Python). Se recomienda utilizar un versión inferior a la 3 (por ejemplo la 2.7.12) para evitar problemas de compatibilidad con NLTK.

Por otro lado, el kit de herramientas de lenguaje natural, o más comúnmente NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN) simbólico y estadísticos para el lenguaje de programación Python. NLTK incluye demostraciones gráficas y datos de muestra. Se acompaña de un libro que explica los conceptos subyacentes a las tareas de procesamiento del lenguaje compatibles el toolkit, además de programas de ejemplo (https://es.wikipedia.org/wiki/NLTK). NLTK se puede descargar desde http://www.nltk.og/, para lo cual deben de seguirse las instrucciones dependiendo del sistema operativo en el que se vaya a trabajar.

Programa base

Con el fin de facilitar el trabajo, partiremos de un programa inicial que habrá de extenderse con términos comunes en un diccionario y expresiones regulares para reconocer la etiqueta morfológica de las palabras contenidas en un texto. Antes de profundizar en el etiquetado morfológico y las expresiones regulares, describimos en esta sección el programa base a extender con el fin de cubrir de antemano los aspectos técnicos.

Una vez instalado python y NLTK debería ser posible ejecutar el programa base “programa_base.py”, cuyo código es el siguiente:

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

import sys
from collections import Counter
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from importlib import reload

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

#Lectura del fichero de texto
f = open ('texto.txt')
freqdist = nltk.FreqDist()
words=nltk.word_tokenize(f.read())
fd = nltk.FreqDist(word.lower() for word in words)
```

```
fdf= fd.most_common(50)

print('Palabras del texto ordenadas por frecuencia')
t=''
for w in fdf:
    t+='('+w[0]+','+str(w[1])+') '
print (t)

dict ={}
dict['.']='PUNT'
dict['la']='DET'
dict['a']='PREP'
dict['para']='PREP'
dict['que']='CONJ'
dict['en']='PREP'
dict['el']='DET'
#Aquí hay se añaden las palabras del diccionario y sus etiquetas

p=[
    (r'.*amos$','VIP1S'),
    (r'.*imos$','VIP1S'),
    (r'.*a$','NCFS'),
    (r'.*$','NCMS'),
    #Aquí hay se añaden los patrones necesarios
    ]

rt=nltk.RegexpTagger(p)
taggedText=rt.tag(words)
for item in taggedText:
 #   if dict.has_key(item[0]):
    if item[0] in dict:
        print (item[0]+' '+dict[item[0]])
    else:
        print (item[0]+' '+item[1])

sys.exit
```

En la primera parte del código se importan las librerías necesarias. En la segunda parte se resuelven aspectos de codificación con el fin de capturar acentos y caracteres no ingleses. Posteriormente, se lee y *tokeniza* el documento de texto. Para simplificar el problema, vamos a minimizar los tipos de etiquetas a generar. El texto a analizar contiene la descripción de algunos conceptos fundamentales en mecánica de coches, con un lenguaje más o menos uniforme.

Tras esto, el código muestra las palabras contenidas en el texto ordenadas por frecuencia. Se puede sustituir el número 50 por un número superior para ver más de ellas. En el siguiente paso, se define un diccionario con los términos más comunes y su etiqueta gramatical correspondiente. Este diccionario debe de ser extendido para aquellas palabras que no puedan ser cubiertas por expresiones regulares según su terminación. En la sección 5.3 del libro de NLTK se ofrecen más detalles sobre como construir y manipular un diccionario en Python.

Posteriormente, la variable “p” incluye el conjunto de expresiones regulares empleadas para generar etiquetas en base a la terminación de la palabra. Nótese que estas expresiones se activan por orden de precedencia, de forma que aquellas palabras que no encajen en ninguna de las expresiones anteriores, activarán la última expresión “*$” que incluye cualquier palabra. Este aspecto es importante e implica que conviene comenzar por las expresiones más específicas con el fin de reducir el número y complejidad de expresiones regulares necesarias. Finalmente, la última sección del código, para cada palabra del texto, aplica el diccionario si éste contiene a la palabra, y en caso contrario aplica las expresiones regulares. En las secciones 3.4 y 5.4 del libro de NLTK se describe con más detalle el uso de expresiones regulares y su aplicación en reconocimiento de etiquetas gramaticales.

El programa base se puede ejecutar mediante la llamada en línea de comandos (en concreto hemos usado python3):

python programa_base.py

y cuyo resultado debería ser (téngase en cuenta que, dado que el código está incompleto, genera etiquetas incorrectas), algo semejante a:

Palabras del texto ordenadas por frecuencia
(de,45) (el,21) (la,20) (que,19) (.,18) (del,17) (los,16) (y,14) (,,13) (a,13) (se,12) (en,10) (motor,10) (para,8) (es,7) (las,7) (al,7) (un,6) (su,5) (vehículo,4) (aceite,4) (bomba,4) (una,4) (culata,3) (uno,3) (son,3) (como,3) (o,3) (colector,3) (filtro,3) (no,3) (presión,3) (con,3) (más,3) (funcionamiento,3) (forma,3) (combustible,3) (inyección,3) (freno,3) (circuito,3) (sistema,3) (asegurar,2) (través,2) (diésel,2) (aire,2) (pedal,2) (cilindros,2) (por,2) (coche,2) (exterior,2) (refrigerante,2) (radiador,2) (bajo,2) (hacia,2) (cuando,2) (inyectores,2) (fricción,2) (elementos,2) (cada,2) (hacer,2) (correcto,2) (tanto,2) (discos,2) (debe,2) (escape,2) (también,2) (nuestro,2) (esenciales,2) (ya,2) (todo,2) (conductos,2) (bolsillo,1) (dos,1) (está,1) (dosificar,1) (condicionará,1) (impurezas,1) (alimentación,1) (medio,1) (hace,1) (barrera,1) (gasolina,1) (habilitados,1) (accede,1) (alrededor,1) (hacen,1) (ritmo,1) (calentamientos,1) (recomendamos,1) (cualquier,1) (ese,1) (vida,1) (este,1) (lo,1) (encarga,1) (lleguen,1) (mantenimiento,1) (transportar,1) (gases,1) (trabajo,1)
El NCMS
filtro NCMS
de NCMS
aceite NCMS
es NCMS
…
etc.

En caso de surgir problemas hasta este punto, como por ejemplo, problemas de codificación dependiendo del sistema operativo, se recomienda consultarlo lo antes posible en el foro del curso virtual.

Etiquetado morfológico en castellano.

En la sección 56.2 se muestra un ejemplo de juego de etiquetas simplificado para el inglés. Sin embargo, el juego de etiquetas empleado en esta práctica será el siguiente:

1. Advervios: ADV
2. Artículos:ART
3. Adjetivos:ADJ
4. Determinantes: DET
5. Pronombres: PRON
6. Conjunciones: CONJ
7. Numerales: NUM
8. Interjecciones: INT
9. Abreviaturas:ABRV
10. Preposiciones: PREP
11. Signos de puntuación: PUNT

Para los verbos y nombres, tomaremos como conjunto de etiquetas gramaticales, las propuestas por el grupo EAGLES para la anotación morfosintáctica de lexicones y corpus para todas las lenguas europeas. Este conjunto de etiquetas se puede consultar en el enlace http://www.cs.upc.edu/~nlp/tools/parole-sp.html.

Para el caso de los nombres, tendremos en cuenta el tipo, género y número. No consideraremos el caso, ni el género semántico ni el grado (ver referencia de EAGLES).

NOMBRES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Nombre	N
2	Tipo	Común	C
		Propio	P
3	Género	Masculino	M
		Femenino	F
		Común	C
4	Número	Singular	S
		Plural	P
		Invariable	N

De forma, que la etiqueta para la palabra “mesa”, sería NCFS. Para los verbos consideraremos directamente el esquema de etiquetado de EAGLE:

VERBOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Verbo	V
2	Tipo	Principal	M
		Auxiliar	A
3	Modo	Indicativo	I
		Subjuntivo	S
		Imperativo	M
		Condicional	C
		Infinitivo	N
		Gerundio	G
		Participio	P
4	Tiempo	Presente	P
		Imperfecto	I
		Futuro	F
		Pasado	S
5	Persona	Primera	1
		Segunda	2
		Tercera	3
6	Número	Singular	S
		Plural	P
7	Género	Masculino	M
		Femenino	F

Es importante tener en cuenta que pueden existir muchos casos de ambigüedad (sección 5.3 del libro base). Por ejemplo, la palabra “partes” podría ser tanto un nombre común plural femenino, como la segunda persona del verbo partir. La resolución de ambigüedad en el etiquetado gramatical, se realiza típicamente mediante métodos basados en reglas (sección 5.4 del libro base) o métodos estocásticos (sección 5.5 del libro base). Éstos métodos tienen en cuenta el contexto de las palabras para su etiquetado. En esta práctica no vamos a abordar el problema de la ambigüedad, pero sí es conveniente que las etiquetas más comunes de entre las ambiguas tengan preferencia en el conjunto de expresiones regulares definidas.

Expresiones regulares

El capítulo 2 del libro base introduce en detalle las expresiones regulares, que es la notación estandar para caracterizar secuencias de texto. Éstas se aplican en todo tipo de sistemas de procesamiento de texto y extracción de información. La sección 3.4 del libro "Natural Language Processing with Python" incluye una descripción detallada de la sintaxis de las expresiones regulares en Python (http://www.nltk.org/book/ch03.html), y la sección 3.7 incluye detalles sobre el uso de expresiones regulares en Python. La sección 4 del capítulo 5 del mismo documento (http://www.nltk.org/book/ch05.html) contiene información de cómo implementar un etiquetador morfológico en Python.

Criterios de evaluación de la práctica

Como indicábamos en la introducción de este documento, la práctica se evaluará en base a los siguientes criterios. El primer criterio es el de **efectividad.** Interesa que en la medida de lo posible, las palabras contenidas en el texto sean etiquetadas correctamente. Esto se podría conseguir simplemente con un diccionario muy exhaustivo. El problema de aplicar esta aproximación es que el sistema no sería capaz de procesar un texto nuevo. El uso de expresiones regulares permite generalizar reglas de etiquetado que pueden aplicarse a nuevas palabras. Por tanto, el segundo criterio de evaluación será el de la **generalidad.** Es decir, en qué medida el sistema es capaz de procesar nuevos textos en base a las reglas de las que dispone. Es importante tener en cuenta que ambos criterios no son siempre compatibles, pero una alta generalidad y simpleza de la aproximación puede compensar la existencia de errores.

En general, conviene definir patrones específicos o entradas en el diccionario para términos muy comunes como determinantes o adverbios, mientras que palabras poco frecuentes deberían de ser cubiertas en la medida de lo posible con patrones más generales.

Antes de la finalización del plazo de entrega, se enviará un texto nuevo para evaluar el sistema. Es importante no realizar ninguna modificación para este nuevo texto, sino ejecutar directamente el sistema. El objetivo es estudiar la dificultad de la generalización de aproximaciones. Para este análisis, debe evaluarse manualmente la tasa de aciertos del sistema en el texto base usado para el desarrollo y en el texto de test, sobre las primeras 50 palabras de cada texto. Es decir, estudiar la efectividad y generalidad del sistema, analizando brevemente las causas más frecuentes de error.

Los aspectos que se evaluarán en esta práctica serán, por orden de prioridad:

1. La calidad del diseño del programa.
2. El análisis de resultados.
3. La efectividad obtenida en ambos textos.

Normas de entrega

La entrega consistirá en tres ficheros comprimidos en un ZIP. Uno de ellos con el **programa implementado** a partir del programa base, otro con las **salidas obtenidas**, y un documento de texto con una breve descripción de la aproximación y el análisis de resultados. Si se considera necesario, se puede añadir algún comentario sobre la metodología seguida o el diseño de las reglas.