Asignatura	Datos del alumno	Fecha
Procesamiento Lenguaje Natural	Apellidos: González Isorna	05/12/2019
	Nombre: Alberto	

Laboratorio 2: Implementación de una tarea de procesamiento del lenguaje natural

Alberto González Isorna

Índice

Asignatura	Datos del alumno	Fecha
Procesamiento Lenguaje Natural	Apellidos: González Isorna	05/12/2019
	Nombre: Alberto	

Introducción

El objetivo de la práctica es utilizar la herramienta de software abierto Natural Language Toolkit (NLTK) para implementar tareas de procesamiento del lenguaje natural en Python.

Descripción de la memoria

El objetivo de la presente memoria es explicar cómo se ha llevado a cabo la resolución de esta.

En este documento se explicará como se ha desarrollado cada uno de los 10 apartados, y se incluirá el código completo a modo de anexo. Igualmente se incluyen en cada apartado capturas de las partes más relevantes del código para su entendimiento y ubicación en el anexo.

El código anexo esta igualmente separado en los mismos apartados que la presente memoria.

El punto de partida de la práctica es una serie de palabras sobre las que se aplicaran las diferentes herramientas de procesamiento de lenguaje natural.

1. Obtener los diferentes synsets

En este primer punto debemos obtener los synsets en WordNet para el conjunto de palabras siguiente:

```
words = ['mosquito','tsetse','housefly','spider','cockroach']
```

Para obtenerlo simplemente hacemos uso de la función synsets(X), siendo X la palabra a encontrar sus synsets. Por lo tanto, para calcular para todas las palabras incluimos un bucle que va recorriendo el vector de palabras expuesto anteriormente.

Asignatura	Datos del alumno	Fecha
Procesamiento Lenguaje Natural	Apellidos: González Isorna	05/12/2019
	Nombre: Alberto	

```
# 1 - Obtén los diferentes synsets para cada una de las siguientes palabras
list_synset = list()
print('Synsets:\n')
words = ['mosquito','tsetse', 'housefly','spider','cockroach']
for i in words:
  list_synset.append(wn.synsets(i))
  print(wn.synsets(i))

d = list()
```

```
Synsets:

[Synset('mosquito.n.01')]
[Synset('tsetse_fly.n.01')]
[Synset('housefly.n.01')]
[Synset('spider.n.01'), Synset('spider.n.02'), Synset('spider.n.03')]
[Synset('cockroach.n.01')]
```

En list_synsets vamos guardando todos los synsets, ya que nos será de utilidad en futuros apartados.

Vemos como todas las palabras tienen una sola agrupación de sinónimos, excepto spider que tiene tres.

2. Definición de cada uno de los sentidos

Para la definición utilizamos un procedimiento similar al apartado anterior, recorremos la lista de palabras aplicando a cada paso la función definition() aplicada al synset. Solo hay que incluir un segundo bucle para tener en cuenta que una palabra puede tener más de un synset.

```
# 2 - Muestra la definición de cada uno de los sentidos o synsets de las palabras
print('\nDefiniciones:')
for i in words:
  sn = wn.synsets(i)
  print('\nWord : ',i)
  d.append(sn[0].definition())
  for k in range(len(sn)):
    print('{} \nDefinition : {}'.format(sn[k],sn[k].definition()))

print('\n\n',d)
```

```
Obtenemos las siguientes definiciones:

Word :  mosquito
Synset('mosquito.n.01')
Definition : two-winged insect whose female has a long proboscis to pierce the skin and
suck the blood of humans and animals

Word :  tsetse
Synset('tsetse_fly.n.01')
```

```
Definition : bloodsucking African fly; transmits sleeping sickness etc.

Word :  housefly
Synset('housefly.n.01')
Definition : common fly that frequents human habitations and spreads many diseases

Word :  spider
Synset('spider.n.01')
Definition : predatory arachnid with eight legs, two poison fangs, two feelers, and
usually two silk-spinning organs at the back end of the body; they spin silk to make
cocoons for eggs or traps for prey
Synset('spider.n.02')
Definition : a computer program that prowls the internet looking for publicly accessible
resources that can be added to a database; the database can then be searched with a search
engine
Synset('spider.n.03')
Definition : a skillet made of cast iron

Word :  cockroach
Synset('cockroach.n.01')
Definition : any of numerous chiefly nocturnal insects; some are domestic pests
```

Como hemos mencionado anteriormente, tenemos una definición única para todas las palabras (el insecto) excepto para spider que pude ser también un programa o un tipo de sartén con patas.

3. Selección de un sentido único

Para la elección del sentido único de sypder de forma automática hemos realizado los siguientes pasos:

3.1. Extraer de la lista que habíamos obtenido con todos los synsets el synset de spyder

3.2. Calcular la similitud de cada palabra de la lista precedente con la palabra spyder

- Para ello, hemos utilizado la función X.path_similarity(Y). Siendo X el synset a comparar con el synset Y.

3.3. Al comparar tendremos por tanto 4 comparaciones por cada sentido de sypder, 12 combinaciones en total. Para tomar una decisión fácilmente hemos calculado la media para cada similitud de las palabras con cada significado de spyder

3.4. Por último, nos hemos quedado con el máximo, es decir, el que de media tiene mas similitud con el resto de las palabras.

Hemos obtenido un resultado coherente, ya que el mas probable es la definición 0. Por extensión hemos incluido el código en el anexo. El resultado es el siguiente:

```
Media de las similitudes :  [0.15476190476190477, 0.046536796536796536, 0.054093567251461985]
Definicion mas acertada  :  Spider 0
Definicion : predatory arachnid with eight legs, two poison fangs, two feelers, and usually two silk-spinning organs at the back
```

4. Sinónimos

Una vez descartado los significados de spider que no tenían relación con el grupo de palabras, nos hemos quedado con un único vector de palabras y synset asociado.

Para calcular los sinónimos de cada palabras hemos recorrido todas las palabras utilizado la función X.lemma_names(), siendo X el synset de una palabra dada.

```
#4) Una vez hayas seleccionado un único sentido para cada una de las palabras,
# para cada uno de esos cinco synsets muestra las palabras que son sinónimos
# y que conforman el synset, es decir los sinónimos que comparten un mismo significado.

# Construimos la lista de sinstets
a = list_sinspider.copy()
a.append([spider[0]])
a

# Vemos los sinonimos
for j in range(len(a)):
  syn = a[j][0]
  print(syn.name(),syn.lemma_names())
```

```
mosquito.n.01 ['mosquito']
tsetse_fly.n.01 ['tsetse_fly', 'tsetse', 'tzetze_fly', 'tzetze', 'glossina']
housefly.n.01 ['housefly', 'house_fly', 'Musca_domestica']
cockroach.n.01 ['cockroach', 'roach']
spider.n.01 ['spider']
```

Vemos por ejemplo como mosquito setsze tiene otros sinónimos como glossina.

5. Hipónimos

Para obtener los hipónimos utilizamos la misma estructura que en aparto anterior pero aplicando la funcion X.hyponyms(), siendo X el synset de una palabra dada.

```
]  # 5) Para cada uno de los cinco synsets obtén las palabras que son hipónimos del sentido de la palabra.
   # Vemos los sinonimos
   for j in range(len(a)):
     syn = a[j][0]
     print(syn.name(),syn.hyponyms())
```

Asignatura	Datos del alumno	Fecha
Procesamiento Lenguaje Natural	Apellidos: González Isorna	05/12/2019
	Nombre: Alberto	

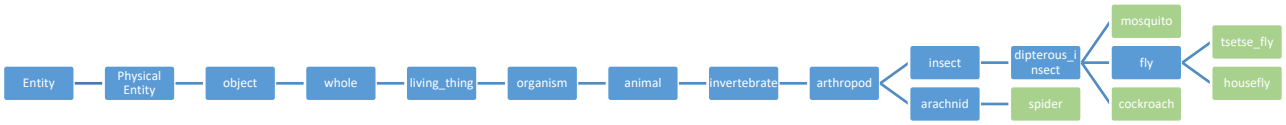
6. Jerarquía Hiperónimos

Para obtener la jerarquía de todos los hiperónimos utilizamos la misma estructura que en aparto anterior pero aplicando la funcion X.hypernym_paths (), siendo X el synset de una palabra dada.

```
for j in range(len(a)):
  syn = a[j][0]
  print(syn.name(),syn.hypernym_paths())
```

7. Dibujo de un tesauro con la Jerarquía de Hiperónimos

Para la jerarquía hemos partido del apartado anterior y subimos en la escala buscando los hiperónimos comunes. El esquema es el siguiente:



8. Ancestro Común más bajo

Para el ancestro común mas bajo nos servimos de la función x.lowest_common_hypernyms(y). Siendo X e Y los synset a calcular el ancestro común mas bajo.

Dado que tenemos que probar todas las posibles combinaciones con los 5 synsets, la primera operación es realizar una lista o set con todas las combinaciones posibles. Esto se hace fácilmente gracias a la función combinations(L,N), siendo L la lista a combinar y N el numero de elementos por combinación.

```
for p in range(len(b)):
  anc,x,y = calcu_ancestro(b[p])
  print('Ancestro Minimo : {} y {} = {}'.format(x,y,anc))
```

```
Ancestro Minimo : Synset('mosquito.n.01') y Synset('tsetse_fly.n.01') = [Synset('dipterous_insect.n.01')]
Ancestro Minimo : Synset('mosquito.n.01') y Synset('housefly.n.01') = [Synset('dipterous_insect.n.01')]
Ancestro Minimo : Synset('mosquito.n.01') y Synset('cockroach.n.01') = [Synset('insect.n.01')]
Ancestro Minimo : Synset('mosquito.n.01') y Synset('spider.n.01') = [Synset('arthropod.n.01')]
Ancestro Minimo : Synset('tsetse_fly.n.01') y Synset('housefly.n.01') = [Synset('fly.n.01')]
Ancestro Minimo : Synset('tsetse_fly.n.01') y Synset('cockroach.n.01') = [Synset('insect.n.01')]
Ancestro Minimo : Synset('tsetse_fly.n.01') y Synset('spider.n.01') = [Synset('arthropod.n.01')]
Ancestro Minimo : Synset('housefly.n.01') y Synset('cockroach.n.01') = [Synset('insect.n.01')]
Ancestro Minimo : Synset('housefly.n.01') y Synset('spider.n.01') = [Synset('arthropod.n.01')]
Ancestro Minimo : Synset('cockroach.n.01') y Synset('spider.n.01') = [Synset('arthropod.n.01')]
```

Asignatura	Datos del alumno	Fecha
Procesamiento Lenguaje Natural	Apellidos: González Isorna	05/12/2019
	Nombre: Alberto	

9. Profundidad

Para la profundidad podríamos partir del apartado siete y contar cuantos saltos hay hasta la palabra deseada o por otro lado utilizar la funcion X.min_depth() siendo X el synset a calcular la profundidad (que es lo que hemos implementado a continuación).

```
# 9) Calcula la profundidad de cada uno de los cinco synsets.
s= 'Profundidad Synset {} \t-> {}'

for j in range(len(a)):
  syn = a[j][0]
  print(s.format(syn.name(),syn.min_depth()))
```

```
Profundidad Synset mosquito.n.01        -> 11
Profundidad Synset tsetse_fly.n.01      -> 12
Profundidad Synset housefly.n.01        -> 12
Profundidad Synset cockroach.n.01       -> 11
Profundidad Synset spider.n.01  -> 10
```

Como vemos coincidimos con el apartado 7, siendo la más baja spider por lo tanto probablemente menos similar con el resto.

10.Similitud de palabras

En este apartado se nos pide que calculemos las similitudes de todas las combinaciones. Para ello aprovechamos la lista echa anteriormente y utilizamos la X.funcion wup_similarity(Y)[1] siendo X e Y los synsets a calcular la similitud. Obtenemos lo siguiente:

[1] Calcula la relación considerando las profundidades de los dos synsets en las taxonomías de WordNet, junto con la profundidad de la LCS (suscriptor menos común). El intervalo va de 0 para palabras nada similares a 1.

```
def calcu_sim(par):
  x = par[0][0]
  y = par[1][0]
  r = x.wup_similarity(y)
  return r,x,y


for p in range(len(b)):
  sim,x,y = calcu_sim(b[p])
  print('Wup similarity : {} y {} \t=> {}'.format(x,y,sim))
```

```
Wup similarity : Synset('mosquito.n.01') y Synset('tsetse_fly.n.01')    => 0.88
Wup similarity : Synset('mosquito.n.01') y Synset('housefly.n.01')      => 0.88
Wup similarity : Synset('mosquito.n.01') y Synset('cockroach.n.01')     => 0.8333333333333334
Wup similarity : Synset('mosquito.n.01') y Synset('spider.n.01')        => 0.782608695652174
Wup similarity : Synset('tsetse_fly.n.01') y Synset('housefly.n.01')    => 0.9230769230769231
Wup similarity : Synset('tsetse_fly.n.01') y Synset('cockroach.n.01')   => 0.8
Wup similarity : Synset('tsetse_fly.n.01') y Synset('spider.n.01')      => 0.75
Wup similarity : Synset('housefly.n.01') y Synset('cockroach.n.01')     => 0.8
Wup similarity : Synset('housefly.n.01') y Synset('spider.n.01')        => 0.75
Wup similarity : Synset('cockroach.n.01') y Synset('spider.n.01')       => 0.782608695652174
```

Como cabía esperar las palabras más similares son las palabras que se relacionan directamente con los insectos voladores: mosquito, mosca tse-tsé y mosca común. Le sigue la cucaracha con el resto y en último lugar la araña que es la que mas se aleja si observamos también la jerarquía del punto 7.

11.Conclusiones

A modo de conclusión cabe mencionar que se ha conseguido satisfactoriamente la ejecución de cada requisito, siendo un punto para mejorar la representación gráfica de la estructura de hiperónimos en Python y el tratamiento mas eficiente de los synsets con funciones encapsuladas (para no tener que hacer un bucle para cada comparación que queramos hacer).

Asignatura	Datos del alumno	Fecha
Procesamiento Lenguaje Natural	Apellidos: González Isorna	05/12/2019
	Nombre: Alberto	

12.Anexos

Código

(ver paginas siguientes)