082057 – Procesamiento del Lenguaje Natural

Profesor: Mg. Ing. Hernán Borré

Lab 4

Semántica léxica (Lexical Semantic)

1 De que se trata?

Este laboratorio se centra en la semántica léxica y utiliza NLTK para brindarte experiencia práctica con **WordNet** y realizar la desambiguación del sentido de las palabras.
El ejercicio se basa en tutoriales en el sitio web de NLTK
http://www.nltk.org/howto/wordnet.html.

2 NLTK environment

El primer paso es importar NLTK y algunos corpus como en los últimos laboratorios. Para llevar a cabo este paso, escriba:

```
>>> import nltk

>>> nltk.download()
```

En el cuadro de diálogo siguiente, elija el libro que descarga todos los corpus utilizados en los ejemplos en el libro NLTK1. Haga clic en descargar y una vez que finalice la descarga, puede cerrar el cuadro de diálogo.

Puede importar los corpus escribiendo:

```
>>> from nltk.book import *
```

3 Accediendo a WordNet

Podes importar WordNet ejecutando:

```
>>> from nltk.corpus import wordnet as wn
```

Como vimos en las clases, WordNet está organizado como **synsets**, cada synset es un conjunto de palabras sinónimas. Una palabra aparecerá en múltiples synsets si tiene múltiples sentidos(senses), así como si puede aparecer en más de un **POS**.

Por ejemplo: sustantivo y verbo. Podes recuperar todos los synsets asociados con una palabra usando la llamada al método synsets():

```
>>>  wn.synsets('dog')
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), Synset('frank.n.02'),
Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]
```

Vemos que hay 7 synsets de sustantivos y 1 synset de verbos asociados con la palabra "perro". Puede restringir la recuperación a un **POS** particular utilizando el argumento pos.

```
>>> wn.synsets('dog', pos=wn.VERB) [Synset('chase.v.01')]
```

As you would have seen in the above examples, a particular synset is referred to using a word, POS tag and a number indicator. As we spoke in class, synsets are ordered according to frequency. dog.n.01 is the first (or most frequent) sense of dog. The second sense of dog can be referred to as dog.n.02. In the output above, the second synset is referred to as frump.n.01. This synset is the same as dog.n.02. frump.n.01 just means this synset is also the first synset for the word 'frump'. Try the following to understand better:

Como habrás visto en los ejemplos anteriores, se hace referencia a un conjunto de sistemas en particular usando una palabra, una etiqueta de POS y un indicador de número.

Como hablamos en clase, los synsets se ordenan según la frecuencia. dog.n.01 es el primer sentido (o el más frecuente) de perro. El segundo sentido de perro se puede denominar dog.n.02.

En la salida anterior, el segundo synset se conoce como frump.n.01. Este synset es el mismo que dog.n.02. frump.n.01 solo significa que este synset es también el primer synset para la palabra "frump". Pruebe lo siguiente para comprender mejor:

```
>>> wn.synset('dog.n.01')

 >>> wn.synset('dog.n.02')

>>> wn.synset('frump.n.01')
```

Ahora, preguntemosle a WordNet definiciones de cada sentido y los sinónimos agrupados en un synset.

```
>>> wn.synset('dog.n.01').definition()
a member of the genus Canis


>>> wn.synset('dog.n.02').definition()
a dull unattractive unpleasant girl or woman
```

Los sinónimos de un synset pueden ser obtenidos usando el método lemmas().

```
>>> wn.synset('dog.n.01').lemmas()

[Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic dog'), Lemma('dog.n.01.Canis familiaris')]
```

Hay tres sinónomos para el synset de 'dog': 'dog', 'domestic dog' y 'Canis familiaris'. Estos aparecen en el ouput de la última instrucción.

Ejercicio: Intenta obtener los lemas para el segundo sentido del "dog".

Los lemmas son….

Las otras Part of Speech son NOUN, ADJ y ADV. Un synset se identifica por un conjunto de 3 partes, siguiendo la forma: **word.pos.nn**

```
>>> wn.synset('dog.n.01')
Synset('dog.n.01')

>>> print(wn.synset('dog.n.01').definition())
a member of the genus Canis (probably descended from the common
wolf) that has been domesticated by man since prehistoric times;
occurs in many breeds

>>> len(wn.synset('dog.n.01').examples())
1

>>> print(wn.synset('dog.n.01').examples()[0])
the dog barked all night

>>> wn.synset('dog.n.01').lemmas()
[Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic_dog'),
Lemma('dog.n.01.Canis_familiaris')]

>>> [str(lemma.name()) for lemma in
wn.synset('dog.n.01').lemmas()]
['dog', 'domestic_dog', 'Canis_familiaris']

>>> wn.lemma('dog.n.01.dog').synset()
Synset('dog.n.01')
```

4 Relaciones entre palabras en WordNet

Ahora veamos cómo consultar las relaciones entre los synsets de WordNet. Recuerde que el hiperónimo se refiere al superconjunto de una entidad y el hipónimo se refiere a subconjuntos más específicos.

Puede consultar hipernimos e hipónimos utilizando los siguientes métodos:

```
>>> dog = wn.synset('dog.n.01')


>>> dog.hypernyms()
[Synset('canine.n.02'), Synset('domestic animal.n.01')]


>>> dog.hyponyms()
Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), Synset('dalmatian.n.02')..
```

Como ya vimos en la sección anterior, las formas de palabras individuales en un conjunto se conocen como lemas. Algunas relaciones solo se mantienen entre lemas (es decir, entre formas de palabras específicas) en lugar de los conjuntos de sinónimos.

Por ejemplo, recupere el primer synset asociado con el adjetivo "good" (bueno, en inglés) e imprima sus lemas.

```
>>> good = wn.synset('good.a.01')

>>> good.lemmas()

[Lemma('good.a.01.good')]
```

Solo hay una forma de lema o palabra en este sintetizador. Puede recuperar antónimos para una forma de palabra de la siguiente manera:

```
>>>g0 = good.lemmas()[0]

>>> g0.antonyms() [Lemma('bad.a.01.bad')]
```

Exploremos también dos relaciones más: meronym y holonym. Un merónimo de una palabra es una subparte o miembro. Un holónimo es un todo del cual la palabra es parte o miembro. Hay métodos separados en NLTK que recuperan merónimos / holónimos para las relaciones entre partes y miembros. P.ej:

```
>>> dog.part meronyms() [Synset('flag.n.07')]
>>> dog.member meronyms() []

>>> dog.part holonyms()
[]
>>> dog.member holonyms() [Synset('canis.n.01'), Synset('pack.n.06')]
```

Este resultado dice que flag.n.07 es parte de dog.n.01 y dog.n.01 es miembro de canis.n.01 y pack.n.06.

Ejercicio: Utilizando lo que ha estudiado hasta ahora, imprima las definiciones de flag.n.07, canis.n.01 y pack.n.06 y vea si ve por qué estos synsets están relacionados de esta manera.

5 Desambiguación de palabras por sentido (Word Sense Desambiguation)

Como vimos en las clases, la tarea de desambiguación del sentido de las palabras es tomar una palabra en el contexto de la oración y mapearla a uno de los sentidos de la palabra, por ejemplo. asignar a un synset en WordNet.

Estudiamos dos enfoques, un sistema de clasificación supervisado donde las palabras de contexto entran como feautres(características) y el otro es el algoritmo de Lesk que utilizó el recurso del diccionario para la desambiguación.

Ejercicio: recuerde el algoritmo de Lesk. ¿Cuáles fueron los pasos?
Usamos el siguiente ejemplo en la clase, intentemos con el mismo.

“The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable- rate mortgage securities.”

(El banco puede garantizar que los depósitos eventualmente cubrirán los costos de matrícula futuros porque invierte en valores hipotecarios de tasa ajustable.)

Queremos eliminar la ambigüedad de “bank” en este contexto.

Ejercicio: ¿Cuántos sentidos crees que tiene la palabra bank? .

Use lo que ha estudiado hasta ahora para recuperar todos los synsets asociados con el "bank" con el POS de sustantivo. ¿Cuántos sentidos sustantivos de "banco" hay en WordNet? .

¿Qué synset es el sentido correcto para la palabra en el contexto de la oración anterior?

Ahora veamos qué nos da el algoritmo Lesk.

```
>>> from nltk.wsd import lesk
>>> from nltk import word tokenize
>>> S = ‘‘The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.’’
>>> S tok = word tokenize(S)
>>> print(lesk(S tok, ’bank’, ’n’))
```

Ejercicio: ¿Qué synset fue producido por Lesk?
¿Entiendes por qué tienes esta predicción? Veamos cómo NLTK está implementando Lesk.

Puede encontrar el código fuente aquí http://www.nltk.org/_modules/nltk/wsd.html.

Hagamos algunos diagnósticos.

```
>>> l = word tokenize((wn.synset(’bank.n.05’).definition()))

>>> m = word tokenize((wn.synset(’bank.n.02’).definition()))

>>> k = set(S tok)


>>> k.intersection(l)

>>> k.intersection(m)
```

¿Ves ahora por qué tienes esta predicción?
¿Qué cambios, de haber alguno, sugeriría para la correspondencia implementada por NLTK?

Ejercicio: para cada ejemplo, a continuación descubra el sentido correcto de WordNet según su criterio. Puede usar la búsqueda en línea de WordNet para navegar a través de los sentidos. http://wordnetweb.princeton.edu/perl/webwn.

I went to the bank to deposit some money.

She created a big mess of the birthday cake.

In the interest of your safety, please wear your seatbelt.

I drank some ice cold water.

Luego use el algoritmo de Lesk para desambiguar las palabras. ¿Cuál es la precisión de la implementación de Lesk de NLTK en estas oraciones?

6 Un poco más allá

A lemma form of a word groups inflections of a word together. ie. 'walked', 'walking', 'walks', 'walk' are all inflections or morphological variants of the word 'walk'. NLTK has facility to lemmatize words.

Si ya estás cómodo con el contenido anterior, podés desafiarte con este ejercicio. Este ejercicio tiene un "final abierto".

Ejercicio: un lemma de una palabra agrupa las inflexiones de una palabra. Es decir, 'walked', 'walking', 'walks', 'walk' son todas inflexiones o variantes morfológicas de la palabra **walk** ("caminar"). NLTK tiene capacidad para lematizar palabras.

```
>>> from nltk.stem import WordNetLemmatizer

>>> wnLemmatizer = WordNetLemmatizer()

>>> wnLemmatizer.lemmatize('dogs', 'n') u'dog'

wnLemmatizer.lemmatize('walking', 'v')
u'walk'
```

Aquí informamos al lematizador si la palabra dada es un sustantivo o un verbo. La configuración predeterminada para el método es un sustantivo, es decir. si corre sin argumentos, el lema será correcto para los sustantivos pero quizás no para los verbos.

```
>>> wnLemmatizer.lemmatize('dogs')

u'dog'
```

Can you improve the Lesk algorithm using a lemmatizer? If so, can you write a function match which takes two strings and returns the matching words between them. We want the matches to be useful for an

algorithm such as Lesk. Maybe you also need to run a simple POS tagger? Use can use an inbuilt POS tagger in WordNet.

¿Podrías mejorar el algoritmo de Lesk con un lematizador? Si es así, ¿podrías escribir una función de match, que tome dos cadenas y devuelva las palabras coincidentes entre ellas? Queremos que las coincidencias(matches) sean útiles para un algoritmo como Lesk. ¿Quizás también necesites usar un etiquetador POS? Podes usar un etiquetador POS incorporado en WordNet.

>>> nltk.pos tag(word tokenize("I went to the bank to deposit some money.")) [('I', 'PRP'), ('went', 'VBD'), ('to', 'TO'), ('the', 'DT'), ('bank', 'NN'), ('to', 'TO'), ('deposit', 'VB'), ('some', 'DT'), ('money', 'NN'), ('.', '.')]

Escribí algunos casos de prueba para tu algoritmo.