Universidad Rey Juan Carlos

TEMA 2 – HOJA DE EJERCICIOS II

Se plantean diferentes ejercicios para utilizar recursos lingüísticos sobre diferentes tareas de procesamiento de lenguaje natural.

Ejercicio 1. Análisis con WordNet

WordNet es un recurso lingüístico para el inglés que organiza las palabras según sus significados. En particular, agrupa las palabras en synsets, definidos como el conjunto de sinónimos que comparten un mismo sentido. Asimismo, describe relaciones y jerarquías entre sentidos, tales como sinonimia, hiperonimia e hiponimia.

Se proponen ejercicios relacionados con el análisis de textos mediante WordNet. Se debe utilizar con NLTK: https://www.nltk.org/api/nltk.corpus.reader.wordnet.html.

Apartado 1.1. Synsets y relaciones en WordNet

a. Obtén los diferentes sentidos de la palabra *fight*. Para cada uno de ellos, imprime su definición, los diferentes lemas asociados al mismo sentido y ejemplos de uso de cada sentido. En la siguiente figura se muestra parte de la salida esperada:

```
Synset: competitiveness.n.01
Def: an aggressive willingness to compete
Lemmas: ['competitiveness', 'fight']
Examples: ['the team was full of fight']

Synset: fight.n.04
Def: an intense verbal dispute
Lemmas: ['fight']
Examples: ['a violent fight over the bill is expected in the Senate']
```

b. Haz lo mismo que en el ejercicio anterior, pero obteniendo los synsets solo cuando la categoría de la palabra sea un verbo. ¿Qué ha cambiado?

c. Obtén todos los sentidos de *bank* cuando es un nombre. ¿Cuántos sentidos tiene? Busca sus hiperónimos y sus hipónimos.

   Recuerda que un hipónimo concreta el significado de su hiperónimo. De esta manera, *mesa* es más específico que *mueble,* y *escritorio* es un tipo particular de *mesa*.

Apartado 1.2. Similitud semántica y desambiguación con WordNet

a. Calcula la similitud semántica entre pares de synsets. En particular, se requiere seleccionar sentidos concretos de las siguientes palabras y mostrar su similitud:

   dog – cat
   vehicle – car
   dog – car
   whale – eagle

   Utiliza diferentes medidas de similitud disponibles en WordNet. Esto incluye:

   1. Medidas basadas en path, donde se mide la longitud que hay en el camino que lleva de un concepto a otro. Cuánto más corto sea el camino, más similares serán.
   2. Medidas basadas en el contenido, de forma que, cuanta más información compartan dos conceptos, más similares serán.

b. Lo más razonable a la hora de desambiguar el sentido de las palabras es tener en cuenta el contexto en el que se encuentran. Uno de los primeros algoritmos desarrollados para abordar este problema es el algoritmo de Lesk. Dada una palabra polisémica y el contexto en el que aparece, este algoritmo devuelve el synset con el mayor número de palabras coincidentes entre la frase en la que se encuentra la palabra a desambiguar y las distintas definiciones de los synsets que puede poseer una palabra.

   Utiliza el algoritmo de Lesk para desambiguar la palabra *bank* en las frases *I went to the bank to deposit my money* y *The land along the river bank has vegetation*. Muestra la definición del synset. ¿Son las acepciones correctas de la palabra? Pista: https://www.nltk.org/howto/wsd.html

Ejercicio 2. Análisis de Sentimiento

Apartado 2.1. Lexicones de sentimiento

Se quiere desarrollar un evaluador de sentimientos basado en el uso de un recurso lingüístico externo. En concreto, se pide utilizar la lista de palabras de opinión disponible en NLTK. Este lexicón asigna un valor de polaridad a cada una de las palabras de dicha lista.

Universidad Rey Juan Carlos

```
from nltk.corpus import opinion_lexicon
print(opinion_lexicon.words())
print(opinion_lexicon.words()[10:20])
print(opinion_lexicon.positive())

print(opinion_lexicon.negative())
```

```
['2-faced', '2-faces', 'abnormal', 'abolish', ...]
['aborts', 'abrade', 'abrasive', 'abrupt', 'abruptly', 'abscond', 'absence', 'absent-minded', 'absentee', 'absurd']
['a+', 'abound', 'abounds', 'abundance', 'abundant', ...]
```

Dadas las siguientes opiniones, indica si son positivas, negativas o neutras:

Opinión 1: *Visceral, stunning and relentless film making. Dicaprio's Herculean, almost purely physical performance and Hardy's wide-eyed intensity coupled with the almost overwhelming beauty of the landscape - those trees, the natural light, the sun peeking through the clouds, rendered the proceedings down to savage poetry. A hypnotic, beautiful, exhausting film.*

Opinión 2: *I saw this film on Friday. For the first 40 minutes involving spoken dialogue they need not have bothered. For me the dialogue was totally unintelligible with grunting, southern states drawl, and coarse accent that made it impossible to understand what they were saying.*

Opinión 3: *It was an idiotic film that produces a magnificent fascination.*

Se puede desarrollar un algoritmo sencillo que busque los tokens del texto de entrada en las listas de palabras positivas o negativas, sumando una unidad si el token se encuentra en la lista de palabras positivas o restando una unidad si está en la lista de palabras negativas. De este modo, se clasificarían las opiniones en función de las siguientes etiquetas.

- **Positiva** si la puntuación global es mayor a 0.
- **Negativa** si la puntuación global es menor a 0.
- **Neutra** si la puntuación global es igual a 0.

Apartado 2.2. Entrenamiento de modelos de aprendizaje automático

Dado el corpus de opiniones de películas disponible en NLTK *movie_reviews*:

a. Divide el conjunto de datos en train y test y obtén el accuracy de clasificación de sentimiento en el conjunto de test con el lexicón utilizado en el apartado anterior.

b. Utiliza el conjunto train del corpus anotado para entrenar un clasificador de opiniones positivas y negativas. En particular, usa NaiveBayesClassifier de NLTK, el cual recibe como entrada un diccionario de características. Puedes basarte en la siguiente función:

```python
def features(tokens: list[str]) -> dict[str, bool]:
    """
    Transforma una lista de tokens en un diccionario de características
    para su uso con clasificadores de NLTK.

    Cada token se convierte en una clave del diccionario con valor True,
    representando su presencia en el documento.

    Parameters
    ----------
    tokens : list[str]
        Lista de tokens del documento.

    Returns
    -------
    dict[str, bool]
        Diccionario donde cada clave es un token y su valor es True.
    """
    return dict([(token, True) for token in tokens])
```

¿Mejoran los resultados respecto al lexicón? ¿Qué ventajas tiene cada enfoque?