Laboratorio: Desambiguación del sentido de las palabras

**Objetivos**

Con este laboratorio el alumno conseguirá aplicar diferentes algoritmos basados en aprendizaje automático supervisado para desambiguar el sentido de las palabras. Además, va a aprender a utilizar la herramienta de *software* abierto *Natural Language Toolkit* (NLTK) con la que implementar tareas de procesamiento del lenguaje natural en Python.

**Descripción**

En este laboratorio debes desarrollar e implementar diferentes algoritmos basados en aprendizaje automático supervisado para desambiguar el sentido de las palabras en Python y utilizando la herramienta de *software* abierto Natural Language Toolkit (NLTK).

Para preparar este laboratorio, simplemente descarga e instala **NLTK 3.3** en tu equipo.

Accede a NLTK 3.3 a través del aula virtual o desde la siguiente dirección web:

<https://www.nltk.org/install.html>

NLTK requiere de Python versiones 2.7, 3.4, 3.5 o 3.6 para funcionar. Por lo que, si no tienes instalado Python, descárgalo e instálalo.

Accede a **Python** a través del aula virtual o desde la siguiente dirección web: <https://www.python.org/downloads/>

Asegúrate de que has instalado NLTK 3.3 adecuadamente antes de la sesión de laboratorio y de revisar el contenido teórico, Semántica léxica y temas anteriores, para tener frescos los diferentes conceptos sobre el procesamiento del lenguaje natural estudiados en esta asignatura.

Durante la sesión del laboratorio debes solucionar un problema sobre desambiguación del sentido de las palabras utilizando el corpus etiquetado en inglés llamado **Senseval 2** y que viene disponible en NLTK.

Accede a más información sobre **Senseval 2** a través del aula virtual o desde la siguiente dirección web:

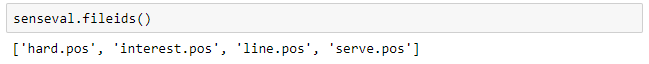
<http://www.nltk.org/howto/corpus.html>

El primer paso es importar el corpus etiquetado utilizando los siguientes comandos:



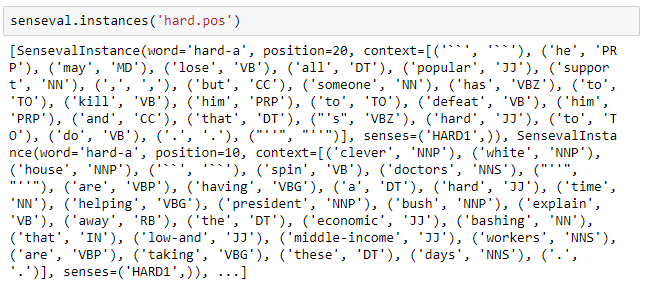
El corpus **Senseval 2** contiene datos etiquetados que sirven para entrenar un clasificador que permita desambiguar el sentido de las palabras. Cada elemento del corpus **Senseval 2** se corresponde con una palabra ambigua. Concretamente en este laboratorio se trabajará con las palabras en inglés «*hard*» y «*serve*», aunque en el corpus hay información de otras dos.

Para poder extraer la información sobre las palabras es imprescindible la manera en la que se identifican en el corpus, es decir, sus identificadores. Con el siguiente comando, se extraen los identificadores de las palabras tratadas en el corpus.



Para cada una de las palabras ambiguas, el corpus contiene una lista de instancias correspondientes a las ocurrencias de esa palabra. Para cada instancia se proporciona la palabra, una lista de sentidos que se aplican a la aparición de esa palabra y el contexto de la palabra.

En la siguiente figura se observa el comando utilizado para visualizar la información que contiene cada instancia de la palabra ambigua «*hard*».



Por ejemplo, en la primera instancia (SensevalInstance) la palabra ambigua (word) es ‘hard-a’, lo que indica que la palabra es ‘hard’ y en este caso la categoría gramatical es un adjetivo, identificado por el sufijo ‘-a’.

El campo position indica la posición en la oración en la que se encuentra la palabra ambigua, en este caso la palabra ‘hard’ se encuentra en la posición 20.

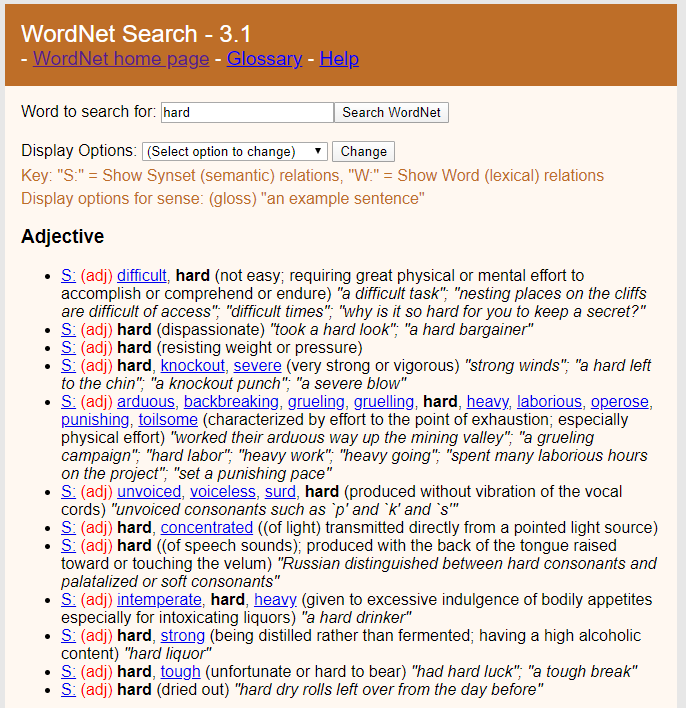
El campo context representa el contexto, es decir, la oración en la que se encuentra la palabra ambigua, en este ejemplo «*he may lose all popular support , but someone has to kill him to defeat him and that 's hard to do trasplantes .* ». El contexto viene representado por pares formados por una palabra y la correspondiente etiqueta gramatical. Por ejemplo, el par (‘he’, ‘PRP’) que aparece en el contexto indica que la categoría gramatical asociada a la palabra ‘he’ es un pronombre personal ‘PRP’.

Por último, el campo senses contiene los posibles sentidos de la palabra ambigua, en el ejemplo ‘HARD1’. Los sentidos del corpus hacen referencia a los sentidos de la palabra recogidos en la base de datos de relaciones léxicas WordNet[[1]](#footnote-2).

En este caso ‘HARD1’ hace referencia la primera definición de la palabra ‘hard’ que aparece en WordNet, a «difícil», «difficult, hard (not easy; requiring great physical or mental effort to accomplish or comprehend or endure)». Esta información se puede obtener utilizando la interfaz de búsqueda web de WordNet cuyo resultado se muestra en la siguiente figura.

Accede al interfaz de búsqueda de **WordNet** a través del aula virtual o desde la siguiente dirección web:

<http://wordnetweb.princeton.edu/perl/webwn>



**Nota**: NLTK implementa también un lector para la información disponible en la base de datos de relaciones léxicas WordNet. **Aunque no es necesario para realizar esta actividad de laboratorio,** WordNet se puede importar utilizando el siguiente comando:

from nltk.corpus import wordnet

En este laboratorio vas a trabajar con algoritmos basados en aprendizaje automático supervisado, por lo tanto, vas a tener que entrenar diferentes clasificadores que permitan desambiguar la palabra ambigua en inglés «*interest*» y y vas a tener que evaluar el desempeño de los clasificadores creados.

Además, vas a experimentar con una aproximación basada en representaciones vectoriales de palabras para resolver la ambigüedad de las palabras.

Las diferentes partes que forman este laboratorio se indican a continuación.

Parte 1: Análisis del corpus

Analiza el corpus Senseval 2 que vas a utilizar para entrenar los clasificadores. Para realizar el análisis utiliza las funcionalidades que aporta NLTK. Desarrolla el código necesario y responde a las siguientes preguntas.

* ¿Cuántos posibles sentidos tienen las palabras ambiguas «*interest*» y «*line*»? ¿Cuáles son esos sentidos? Para cada sentido indica la etiqueta que aparece en el corpus.
* ¿Cuántas instancias hay en el corpus para cada uno de los sentidos de las palabras ambiguas «*interest*» y «*line*»? Es decir, cuantas oraciones hay en el corpus etiquetadas con cada uno de los sentidos.
* En el contexto, las palabras ambiguas pueden aparecer en diferentes formas gramaticales. Por ejemplo, en el caso de la palabra ambigua «hard», esta aparece tanto la forma base, el adjetivo «hard» como en comparativo «harder» y como en superlativo «hardest». ¿Qué formas gramaticales aparecen en el contexto para cada una de las palabas ambiguas «*interest*» y «*line*»?
* ¿Tienen todas las instancias que forman el corpus el formato que se ha descrito anteriormente? Si hay alguna instancia que no cumpla con ese formato, indica cuales serían las incongruencias que presenta y muestra algunos ejemplos.

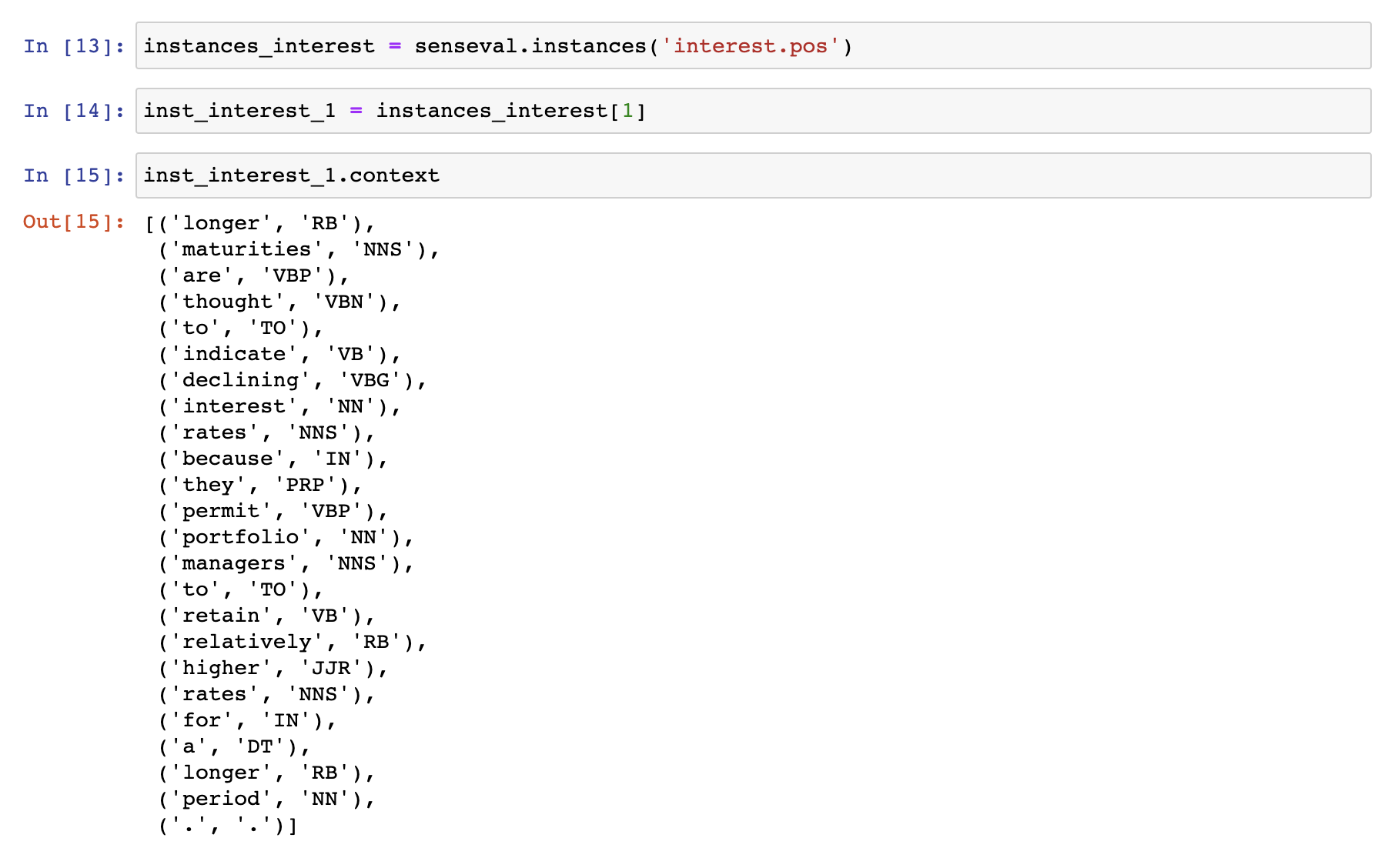
Parte 2: Extracción de características

Para poder entrenar un clasificador es necesario extraer un conjunto de características lingüísticas a partir del corpus etiquetado. Por lo tanto, debes crear el código en Python que te permita extraer diferentes conjuntos de características a partir de Senseval 2.

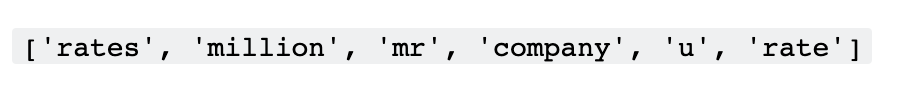
**Extracción de características basada en las palabras vecinas**

Debes extraer un **conjunto de características basado en las palabras vecinas**. Para una instancia del corpus, debes desarrollar el código que sea capaz de extraer el vector de características que indican si las palabras de un vocabulario, que se debe construir previamente, aparecen o no en el contexto (la oración completa en que aparece la palabra ambigua).

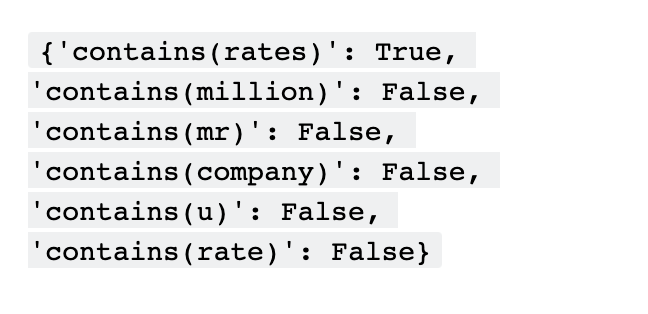
Para una instancia de la palaba ambigua «interest» su contexto se muestra a continuación:



Suponiendo que el vocabulario usado para crear extraer las características es el siguiente:



Entonces el vector de características extraídas para esa instancia sería:



Este vector de características indica que en el contexto de la palabra ambigua aparece la palabra *«rates»* y no aparecen las palabras *«million»*, *«mr»*, *«company»*, *«u»* y *«rate»*.

Para extraer el vector de características basado en las palabras vecinas debes seguir los siguientes pasos:

1. **Construcción del vocabulario o *bags of words*.** Como ya se ha indicado, para poder obtener el vector de características, se debe construir previamente un vocabulario. Para cada una de las palabras del vocabulario, se debe consultar si la palabra aparece en el contexto de la palabra ambigua. Si la palabra del vocabulario aparece en el contexto entonces en el vector de características aparecerá True para esa palabra y si no, False.

Tomemos como ejemplo un vocabulario creado sobre el que se ha construido el vector de características. Este vocabulario es ['rates', 'million', 'mr', 'company', 'u', 'rate']. Este vocabulario se usará posteriormente para construir el vector de características del ejemplo que usamos para darte una orientación.

* **¿Cómo construyo mi vocabulario o bags of words?** Lo que debes hacer para tu entrega de este laboratorio es utilizar como vocabulario las **m** palabras más frecuentes que aparecen en las instancias que conforman el conjunto de datos, es decir en las oraciones que contienen las palabras ambiguas y que forman parte del corpus. Entonces, para crear el bag ***of words*** (bolsa de palabras) debes extraer el conjunto de las n palabras más frecuentes. Para ello te puedes ayudar de la función nltk.FreqDist() que proporciona información sobre la distribución de frecuencias de las palabras que aparecen en un texto.
* Cuando obtengas las palabras más frecuentes, debes eliminar los signos que puntuación y las palabras vacías (aquellas sin significado como artículos, pronombres o preposiciones, las llamadas *stop words* en inglés). También debes eliminar las diferentes formas gramaticales de la palabra ambigua, por ejemplo, para desambiguar la palabra «*hard*» no tendría sentido utilizar la palabra «*harder*» ni la palabra «*hardest*». Esto debes adecuarlo a la palabra ambigua *«interest»*.
* Para la eliminación del conjunto de palabras no útiles del vocabulario se puede usar un código parecido al que se indica a continuación. Debes tener en cuenta que **en este código faltaría añadir las palabras que has identificado en la Parte 1** de este laboratorio como las diferentes formas gramaticales de las palabras ambiguas.

from nltk.corpus import stopwords

import string

OTHER\_WORDS = ["''", "'d", "'ll", "'m", "'re", "'s", "'t", "'ve", '--', '000', '1', '2', '3', '4', '5', '6', '8', '10', '15', '30', 'I', 'F', '``', 'also', "don'", 'n', 'one', 'said', 'say', 'says', 'u', 'us']

STOPWORDS\_SET =

set(stopwords.words('english')).union(set(string.punctuation),

set(OTHER\_WORDS))

* **Ejemplo de vocabulario para un tamaño de 6**. Por ejemplo, si se quiere entrenar un clasificador que permita identificar los diferentes sentidos de la palabra «*hard*» y se utilizan para entrenar y validar el modelo las instancias etiquetadas para esta palabra, la bolsa de palabras en el caso de considerar las seis palabras más frecuentes (m=6) sería la presentada anteriormente ['time', 'would', 'get', 'work', 'find', 'make']. Mientras que para la palabra «interest» sería la presentada anteriormente ['rates', 'million', 'mr', 'company', 'u', 'rate'].

1. **Construcción del conjunto de características basado en palabras vecinas.** Utiliza un diccionario en Python para guardar el conjunto de características. La clave del diccionario serán las palabras del vocabulario y el valor debe ser un *booleano* para indicar la aparición o no de las palabras en el contexto. Por ejemplo, en el vector de características {'contains(rates)': True, 'contains(million)': False, 'contains(mr)': False, 'contains(company)': False, 'contains(u)': False, 'contains(rate)': False} una de las claves del diccionario es 'contains(rates)' y su valor es True lo que indica que en el contexto de la palabra ambigua aparece la palabra *«rates»*..

* Debes mostrar el vector de características resultante para una de las instancias del corpus.
* **Importante:** En el cómputo del vector de características basado en las palabras vecinas debes utilizar como contexto la oración completa donde aparece la palabra ambigua. Es decir, todas las palabras que forman la oración guardada en el campo context de la instancia.

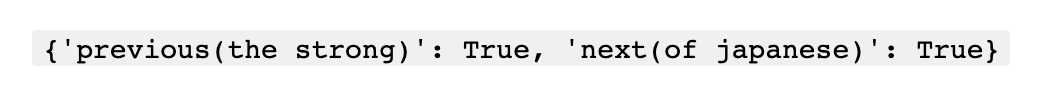
**Extracción de características basada en características de colocación**

Debes extraer también un **conjunto de características de colocación**. Para una instancia del corpus, debes desarrollar el código que sea capaz de extraer el vector de características formado por la secuencia de *n* palabras que ocurren antes de la palabra ambigua y la secuencia de *n* palabras que ocurren después de la palabra ambigua, los llamados n-gramas.

Para una instancia de la palaba ambigua «*interest*» su contexto se muestra a continuación:



Entonces el vector de características de colocación para el bigrama anterior y posterior sería:



Este vector de características indica que antes de la palabra ambigua se encuentran las palabras *«the strong»* y después de la palabra ambigua las palabras *«of japanese»*.

* **¿Cómo construyo mi conjunto y que n utilizo?** Utiliza un diccionario en Python para guardar el conjunto de características, la clave del diccionario debe indicar la secuencia de palabras de contexto y si aparecen antes o después de la palabra ambigua y el valor asociado a la clave debe ser un booleano verdadero. Usaremos **n=2.**  
  Por ejemplo, en el vector de características {'previous(the strong)': True, 'next(of japanese)': True} una de las claves del diccionario es 'previous(the strong)' y su valor es True lo que indica que antes de la palabra ambigua se encuentran las palabras *«the strong»*. En este caso, al tener secuencias de dos palabras (*n=2*), se están considerando bigramas y la ventana tendría tamaño cinco (*2n+1*). Por lo tanto, si la palabra ambigua es \*«interest» en el contexto guardado en el campo context de la instancia, aparece la siguiente parte de la frase «the strong interest of japanese».
* Debes mostrar el vector de características resultante para una de las instancias del corpus.
* **Importante:** Debes tener en cuenta los posibles casos en los que la palabra ambigua aparezca al principio o final de la frase, ya que en esas instancias no vas a poder obtener una secuencia de palabras de longitud n. Por ejemplo, para la instancia cuyo contexto es: […('should', 'MD'), ('be', 'VB'), ('refunded', 'VBN'), (',', ','), ('plus', 'CC'), ('interest', 'NN'), ('.', '.'), …], si n=2 deberías obtener el siguiente vector de características: {'previous(, plus)': True, 'next(.)': True}.
* **Nota**: aunque no es imprescindible para realizar esta actividad de laboratorio, puedes utilizar las funcionalidades para trabajar con n-gramas que ofrece NLTK. Estas se pueden importar utilizando el siguiente comando:

from nltk import ngrams

Parte 3: Entrenamiento de clasificadores

Debes entrenar diferentes clasificadores que permitan desambiguar la palabra ambigua en inglés «*interest*». Además, vas a tener que evaluar el desempeño de los clasificadores creados. Por lo tanto, debes crear el código en Python que te permita entrenar estos clasificadores y evaluarlos.

El tipo de clasificador que vas a utilizar en este laboratorio es el *Naive Bayes*. Para importar el clasificador y el paquete que te permita evaluar su rendimiento debes utilizar el siguiente comando:



Una vez hayas importado los paquetes anteriores, para entrenar un clasificador *Naïve Bayes* puedes usar el comando NaiveBayesClassifier.train() y para evaluarlo accuracy(). Además, puedes utilizar el clasificador entrenado para clasificar una instancia utilizando su método classify(). Por último, puedes obtener la matriz de confusión utilizando el comando nltk.ConfusionMatrix().

Para realizar esta parte del laboratorio debes seguir los siguientes pasos:

1. **Entrenamiento de dos clasificadores para la palabra «interest».** Debes entrenar dos clasificadores que permitan desambiguar la palabra «*interest*», es decir, que debes entrenar los clasificadores utilizando las instancias disponibles en el corpus Senseval 2 para esta palabra ambigua.

* **Conjuntos de entrenamiento y test**. Para entrenar y validar divide las instancias disponibles en una proporción del 80-20 % y recuerda que en el conjunto de datos de entrenamiento deben aparecer instancias de todas las clases.
* **Clasificador basado en las** **palabras vecinas.** Con el conjunto de datos de entrenamiento, entrena un clasificador para «*interest*» que use como características el conjunto basado en las **palabras vecinas** cuyo código has implementado en la parte 2 de este laboratorio. Para definir el vocabulario utiliza las **250** palabras más frecuentes (***m=250***).
* **Clasificador basado en características de colocación**. Con el conjunto de datos de entrenamiento, entrena un clasificador para «*interest*» que use como conjunto de características las de colocación cuyo código has implementado en la parte 2 de este laboratorio. Para definir la **ventana de contexto** utiliza la secuencia de dos palabras que ocurren antes de la palabra ambigua y la secuencia de dos palabras que ocurren después de esta **(n=2).**

1. **Validación de los clasificadores para la palabra «interest».** Utilizando el conjunto de datos de test que has generado previamente, obtén la exactitud (*accuracy*) y la matriz de confusión para cada uno de los dos clasificadores que permiten desambiguar el sentido de la palabra «*interest*».

* Debes mostrar la exactitud (*accuracy*) y la matriz de confusión resultantes de la validación de cada uno de los dos clasificadores.

Debes comparar y analizar los resultados de rendimiento de los clasificadores. Para ello responde a las siguientes preguntas:

* ¿Cuál es el conjunto de características que aporta mejores resultados? ¿Por qué?
* ¿Cuál es el sentido más difícil de identificar? ¿Por qué?
* ¿Qué posibles mejoras se podrían aplicar para mejorar el rendimiento de los clasificadores creados? No es necesario que las implementes, solo que las comentes.

1. **Instancias clasificadas incorrectamente para «*interest*».** Para el clasificador que permite desambiguar la palabra «interest» y que utiliza las características de colocación, obtén las instancias que pertenecen al sentido ‘interest\_1’ y que se han clasificado incorrectamente.

* Presenta en el informe la oración en la que aparece la palabra ambigua (el contexto) para cada una de esas instancias y la etiqueta en la que han sido erróneamente clasificadas.

1. **Análisis de resultados del rendimiento de los clasificadores**. Presenta en el informe una tabla resumen con los valores de exactitud para cada uno de los 2 clasificadores que has entrenado previamente y responde a las siguientes preguntas:

* En Aprendizaje automático es habitual comparar resultados contra un baseline simple que asigna una clase con la probabilidad 1/n (siendo n el número de clases en las que se clasifica) o bien asignando una clase teniendo en cuenta su probabilidad de aparición en el corpus. Compara la exactitud de los clasificadores que has implementado con la que proporcionaría un clasificador que asignara el sentido de forma aleatoria. ¿Cuál sería el mejor clasificador?

Parte 4: Extracción de características utilizando vectores de representación de palabras (word2vec)

En este apartado de la práctica, te proponemos que completes la implementación de una representación de palabras utilizando el método word2vec.

word2vec es un método de representación de palabras en vectores que captura información morfosintáctica y semántica de la palabra mediante el análisis de las frecuencias de los contextos de aparición de las palabras generando representaciones vectoriales de las palabras.

Para ello utilizaremos la implementación de este método de la librería *spacy*. Su documentación la podéis encontrar en el siguiente [enlace](https://spacy.io/usage/linguistic-features#vectors-similarity).  
Descargamos el modelo de vectores word2vec más pequeño disponible en inglés para poder hacer nuestras pruebas.

Con este modelo se puede obtener la representación vectorial de una palabra en concreto.

Pero hay que tener en cuenta que el vocabulario del modelo es limitado y no todas las palabras están incluidas.

Si pudiéramos representar de manera vectorial las definiciones de los diferentes sentidos parece razonable pensar que el vector del sentido más adecuado para la instancia de nuestra palabra ambigua será aquel más próximo a la representación vectorial de la instancia.

Así, queremos tener una función que dada una instancia donde aparezca una palabra ambigua y su conjunto de definiciones nos diga qué sentido o sentidos son los más cercanos al sentido de la instancia. Utilizaremos la similitud coseno para medir esa cercanía de sentidos.

* Debes completar la función que dada una instancia de una palabra ambigua devuelve las definiciones de los sentidos de la palabra ambigua ordenadas de más a menos similitud entre ellas.
* Una vez completada, aplica tu función de desambiguación sobre las instancias 1 y 10 de la palabra ambigua *interest*, mostrando hasta los 3 sentidos más cercanos a las mismas. Utiliza las salidas de las instancias y las definiciones como apoyo para analizar los resultados obtenidos.
* ¿Qué tasa de acierto has observado en estos ejemplos si consideras el sentido con mayor similitud? ¿Y si consideras los 3 primeros?
* ¿A qué se pueden deber los fallos de predicción mediante la similitud?
* ¿Qué ventajas e inconvenientes presenta el método basado en vectores de palabras frente a los métodos de clasificación basado en representaciones de características?

Parte 5: Conclusiones sobre el uso de aprendizaje automático supervisado para desambiguar el sentido de las palabras

Una vez hayas implementado diferentes clasificadores para desambiguar el sentido de diferentes palabras y analizado su desempeño, reflexiona sobre el uso de algoritmos basados en aprendizaje automático supervisado para resolver la tarea de desambiguación del sentido de las palabras. Para ello responde de forma razonada a las siguientes preguntas:

* ¿Cuáles son las limitaciones de los clasificadores que has creado para la desambiguación del sentido de las palabras?
* ¿Qué alternativas propondrías para superar esas limitaciones y obtener algoritmo que resuelva mejor el problema de la desambiguación del sentido de las palabras?

Criterios de evaluación

* Debe resolver la actividad en el Jupyter Notebook proporcionado junto con este enunciado. Si no se usa el archivo proporcionado la actividad será calificada con cero puntos.
* Para la evaluación de la actividad se debe entregar el **Jupyter Notebook** que contenga:
  + El código en Python que permita resolver la actividad, debidamente comentado.
  + Todos los resultados que se solicita en el enunciado y que genera el código implementado.
  + Las respuestas a las preguntas planteadas en el enunciado.
* Se valorarán positivamente, los comentarios del código claros y oportunos, que permitan entender las decisiones de implementación realizadas. Si el código no está comentado de forma suficiente, se penalizará en la calificación de la actividad.
* Se valorará positivamente, la respuesta clara, breve y bien argumentada a las preguntas.
* Si se presenta el código, pero no se muestran los resultados que éste genera, no se considerarán esos resultados en la calificación de la actividad.
* Si se detecta **plagio** de cualquier fuente (internet…), ya sea en el código, en los comentarios o en las respuestas, el alumno obtendrá una calificación para la actividad de cero puntos.
* Si se detecta **copia** entre alumnos en el código, en los comentarios o en las respuestas, todos los alumnos involucrados obtendrán una calificación para la actividad de cero puntos.

**Extensión máxima:** no hay restricciones en la extensión.

|  |  |  |  |
| --- | --- | --- | --- |
| Laboratorio: Desambiguación del sentido de las palabras (valor real: 5 puntos) | Descripción | Puntuación máxima  (puntos) | Peso  % |
| Análisis del corpus | El código para analizar el corpus es correcto y las respuestas a las preguntas sobre el corpus son correctas. | 1 | 10% |
| Extracción de características | El código para extraer dos conjuntos diferentes de características a es correcto. | 2 | 20% |
| Código para el entrenamiento clasificadores | El código para entrenar los clasificadores, evaluar su rendimiento y presentar los errores de clasificación es correcto. | 2 | 20% |
| Reflexiones sobre el entrenamiento de clasificadores | Las respuestas a las preguntas sobre el rendimiento de los diferentes clasificadores entrenados para desambiguar son correctas. | 1 | 10% |
| Aproximación vectorial para la desambiguación | El código para utilizar un modelo de vectores y la similitud coseno para la desambiguación del sentido de las palabras es correcto. | 2 | 20% |
| Reflexiones sobre la implementación vectorial | Las respuestas a las preguntas sobre el uso de la aproximación vectorial para la desambiguación del sentido de las palabras son correctas. | 1 | 10% |
|  |  | **10** | **100 %** |

1. Puede que los sentidos que aparecen en Senseval 2 difieran de los que se encuentran actualmente en WordNet, debido a la constante actualización de este. En este laboratorio no será necesario trabajar con WordNet, se menciona como información adicional. [↑](#footnote-ref-2)