Actividad autocorregible: Desambiguación del sentido de las palabras

Objetivos

Con esta actividad, 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.

Pautas de elaboración

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

Para preparar esta actividad, debes descargar e instalar **NLTK 3.3** en tu equipo. Sigue las instrucciones de instalación que se indican en el enlace.

Accede a la página web a través del aula virtual o desde la siguiente dirección: <https://www.nltk.org/install.html>

NLTK requiere de Python versiones 3.6, 3.7, 3.8, o 3.9 para funcionar. Por lo que, si no tienes instalado Python, descárgalo e instálalo desde el siguiente enlace.

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

Asegúrate de que has instalado NLTK 3.3 adecuadamente antes de comenzar con la actividad y la sesión de presentación.

Durante la realización de la actividad debes solucionar un problema sobre desambiguación del sentido de las palabras utilizando el corpus etiquetado en inglés llamado **Senseval 2** y que está disponible con NLTK. Puedes acceder a más información sobre Senseval 2 desde el siguiente enlace.

Accede a la página web a través del aula virtual o desde la siguiente dirección: <http://www.nltk.org/howto/corpus.html>

El primer paso es importar el corpus etiquetado utilizando las siguientes órdenes:



Figura 1. Importación del corpus etiquetado en NLTK.

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 la siguiente orden, se extraen los identificadores de las palabras tratadas en el corpus.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Figura 2. Identificador de las palabras 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 la orden utilizada para visualizar la información que contiene cada instancia de la palabra ambigua «*hard*».

Texto, Carta

Descripción generada automáticamente

Figura 3 Visualización de las instancias de *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, 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.

En este caso, ‘HARD1’ hace referencia la primera definición de la palabra ‘hard’ que aparece en WordNet, «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 donde podrás indicar las palabras que quieres consultar. Para acceder a la interfaz de búsqueda de WordNet usa el siguiente enlace.

Accede a la página web a través del aula virtual o desde la siguiente dirección: <http://wordnetweb.princeton.edu/perl/webwn>

En la siguiente figura se muestran los resultados en WordNet para la palabra «*hard*».

Texto

Descripción generada automáticamente

Figura 4. Búsqueda de «*hard*» en WordNet.

**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 la siguiente orden:

from nltk.corpus import wordnet

En esta actividad vas a trabajar con algoritmos basados en aprendizaje automático supervisado, por lo tanto, vas a tener que entrenar diferentes clasificadores que permitan desambiguar las palabras ambiguas en inglés «*hard*» y «*serve*», y vas a tener que evaluar el desempeño de los clasificadores creados.

Las diferentes partes que forman esta actividad se indican a continuación. En cada parte encontrarás la descripción de todos los pasos e implementaciones que debes realizar y marcado en azul, y las preguntas que debes responder.

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 que se marca en el *notebook* **Actividad\_3.ipynb,** que te servirá de base para realizar la actividad. Puedes responder a las siguientes preguntas para profundizar en la comprensión del ejercicio:

* ¿Cuántos posibles sentidos tienen las palabras ambiguas «*hard*» y «**serve**»? ¿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 «*hard*» y «*serve*»? Es decir, ¿cuántas 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*», 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 «*hard*» y «*serve*»?
* ¿Tienen todas las instancias que forman el corpus el formato que se ha descrito anteriormente?

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.

Parte 2. A. Extracción de características basada en las palabras vecinas

* Repaso de conceptos.

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 «*hard*» su contexto se muestra a continuación:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 5. Contexto de una instancia de la palabra «*hard*».

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

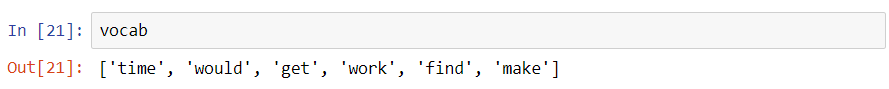


Figura 6. Ejemplo de un vocabulario (*bag of words*).

Entonces el vector de características extraídas para esa instancia sería:

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

Figura 7. Vector de características para la instancia mostrada en la figura 5.

Este vector de características indica que en el contexto de la palabra ambigua aparece la palabra «time» y no aparecen las palabras «would», «get», «work», «find» y «make».

Pasos para la extracción de características basada en palabras vecinas

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.

* + ¿Cómo construyo mi vocabulario o *bags of words*? Lo que debes hacer en la actividad 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 la *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*».
  + 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.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Figura 8. Construcción del conjunto de palabras STOPWORDS.

* + 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'].

1. Construcción del conjunto de características basado en palabras vecinas. Utiliza un diccionario en Pythonpara 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(time)': True, 'contains(would)': False, 'contains(get)': False, 'contains(work)': False, 'contains(find)': False, 'contains(make)': False} una de las claves del diccionario es 'contains(time)' y su valor es True lo que indica que en el contexto de la palabra ambigua aparece la palabra «time».

Suponiendo que el vocabularioes ['time', 'would', 'get', 'work', 'find', 'make'].

**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.

Parte 2. B. 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 «*hard*» su contexto se muestra a continuación:

Texto

Descripción generada automáticamente con confianza media

Figura 9. Contexto para una instancia de «hard».

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



Figura 10. Vector de características de colocación para la instancia de la figura 8.

Este vector de características indica que antes de la palabra ambigua se encuentran las palabras «*have a*» y después de la palabra ambigua las palabras «*time imagining*».

* ¿Cómo construyo mi conjunto y qué 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, el vector de características {'previous(have a)': True, 'next(time imagining)': True} una de las claves del diccionario es ' previous(have a)' y su valor es True lo que indica que antes de la palabra ambigua se encuentran las palabras «have a». 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 «*hard*» en el contexto guardado en el campo context de la instancia, aparece la siguiente parte de la frase «*have a hard time imagining*».

**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: [('some', 'DT'), ('hard', 'JJ'), ('choices', 'NNS'), ('had', 'VBD'), ('to', 'TO'), ('be', 'VB'), ('made', 'VBN'), …] si *n=2* deberías obtener el siguiente vector de características: {'previous(some)': True, 'next(choices had)': True}.

Parte 3. Entrenamiento de clasificadores

Debes entrenar diferentes clasificadores que permitan desambiguar las palabras ambiguas en inglés «*hard*» y «*serve*». 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:



Figura 11. Importación de las bibliotecas para construir el clasificador y evaluar su rendimiento.

Entrenamiento y evaluación

Una vez hayas importado los paquetes anteriores, para entrenar un clasificador *Naive Bayes* puedes usar el comando NaiveBayesClassifier.train() y para evaluarlo accuracy().

Uso del clasificador y obtención de la matriz de confusión

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 de la actividad, debes seguir los siguientes pasos:

* Entrenamiento de dos clasificadores para la palabra «*hard*».

Debes entrenar dos clasificadores que permitan desambiguar la palabra «*hard*», es decir, que debes entrenar los clasificadores utilizando las instancias disponibles en el corpus Senseval 2 para esta palabra ambigua. Uno estará basado en la extracción de características basadas en palabras vecinas y el otro en las características de colocación.

Los pasos que se deben realizar son:

* + 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 «*hard*» 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 «*hard*» 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).
  + Validación de los clasificadores para la palabra «*hard*». 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 «*hard*».

Debes analizar la exactitud (*accuracy*) y la matriz de confusión resultantes de la validación de cada uno de los dos clasificadores. Además, debes comparar y analizar los resultados de rendimiento de los clasificadores. Para ello, puedes tratar de responder a las siguientes preguntas a modo de guía.

¿Cuál es el conjunto de características que aporta mejores resultados? Justifica tu respuesta.

¿Cuál es el sentido más difícil de identificar? Justifica tu respuesta.

¿Qué posibles mejoras se podrían aplicar para mejorar el rendimiento de los clasificadores creados? No es necesario que las implementes, solo que las comentes.

* + Instancias clasificadas incorrectamente para «*hard*».Para el clasificador que permite desambiguar la palabra «*hard*» y que utiliza las características de colocación, obtén las instancias que pertenecen al sentido ‘HARD1’ y que se han clasificado incorrectamente.
* Entrenamiento y validación de dos clasificadores para la palabra «*serve*».Repite el proceso anterior para entrenar y validar dos clasificadores que permitan desambiguar la palabra «*serve*». Puedes aprovechar el código que has generado anteriormente.
  + Crea los conjuntos de entrenamiento y test para las instancias donde la palabra ambigua es «*serve*». Mantén la proporción del 80-20 % para la creación de los conjuntos de entrenamiento y de test.
  + Entrena un clasificador para «*serve*» que use como características el conjunto basado en las palabras vecinas. Para definir el vocabulario, utiliza las 250 palabras más frecuentes (*m=250*).
  + Entrena un clasificador para «*serve*» que use como conjunto de características las de colocación. 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).
  + Obtén el valor, la exactitud (*accuracy*) para cada uno de los dos clasificadores que permiten desambiguar el sentido de la palabra «*sense*».
* Análisis de resultados del rendimiento de los clasificadores.

Puedes generar una tabla resumen con los valores de exactitud para cada uno de los cuatro clasificadores (dos para cada palabra ambigua) que has entrenado previamente y tratar de responder a las siguientes preguntas:

* + ¿Por qué no es justo comparar directamente la exactitud aportada por los clasificadores que han aprendido diferentes palabras ambiguas?
  + ¿Cómo podrías hacerlo para que la comparación entre clasificadores que desambiguan palabras diferentes tenga sentido?
  + Compara la exactitud de los clasificadores con la que proporcionaría un clasificador que asignara el sentido de forma aleatoria. ¿Cuál sería el mejor clasificador tomando como referencia (*baseline*), el clasificador aleatorio?
  + ¿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 un algoritmo que resuelva mejor el problema de la desambiguación del sentido de las palabras?

Preguntas

1. ¿Qué versiones de las palabras ambiguas aparecen para HARD y SERVE en el corpus utilizado en la tarea?

A. HARD: hardest, harder, hard, hardening. SERVE: serve, serving, served, serves

B. HARD: hardest, hard, harder. SERVE: serve, serving, served, serves

C. HARD: hardest, harder, hard, hardening. SERVE: serve, serving, serves

D. HARD: hardest, harder, hard. SERVE: serve, serving, serves

1. ¿Cuántos posibles sentidos tiene la palabra ambiguas «serve»? ¿Qué POS Tags aparecen asociados a ellos?

A. SERVE1, SERVE2, SERVE3. POS Tags: VBG, VB, VBD, VBZ

B. SERVE8, SERVE12, SERVE15. POS Tags: VBG, VB, VBD, JJ

C. SERVE1, SERVE2, SERVE3. POS Tags: JJ

D. SERVE2, SERVE6, SERVE10, SERVE12. POS Tags: VBG, VB, VBD, VBZ

1. ¿Cuál de estas afirmaciones es correcta sobre las características basadas en las palabras vecinas que se han construido en la tarea para entrenar el modelo de clasificación?

A. Recogen información sobre las m palabras más frecuentes que aparecen en las instancias que conforman el conjunto de datos. No se conserva la información sobre el orden de las palabras en el texto.

B. Recogen información sobre 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. Se conserva la información el orden de las palabras en el texto.

C. Ambas son correctas.

D. Ninguna de las anteriores.

1. ¿Cuál de estas afirmaciones es correcta sobre las características de colocación que se han construido en la tarea para entrenar el modelo de clasificación?

A. Recogen información sobre las m palabras más frecuentes que aparecen en las instancias que conforman el conjunto de datos. No se conserva la información sobre el orden de las palabras en el texto.

B. Recogen información sobre 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. Se conserva la información el orden de las palabras en el texto.

C. Ambas son correctas.

D. Ninguna de las anteriores.

1. ¿Cuál es el sentido más difícil de clasificar para el caso de HARD y por qué?

A. El sentido más difícil de clasificar es HARD1, ya que hay un problema de desbalanceo de datos. HARD3 y HARD2 aparecen muchas más veces que HARD1, por lo que el modelo tiene más dificultades para predecir ese caso.

B. El sentido más difícil de clasificar es HARD2, ya que hay un problema de desbalanceo de datos. HARD1 y HARD2 aparecen muchas más veces que HARD3, por lo que el modelo tiene más dificultades para predecir ese caso.

C. El sentido más difícil de clasificar es HARD3 (cerca de HARD2), ya que hay un problema de desbalanceo de datos. HARD1 aparece muchas más veces que HARD3 (y que HARD2), por lo que el modelo tiene más dificultades para predecir ese caso.

D. La dificultad para clasificar los 3 sentidos es parecida, ya que los datos por cada una de las clases están balanceados.

1. ¿Cuál de estas afirmaciones es verdadera sobre el modelo de clasificación para la desambiguación del sentido de las palabras visto en la tarea?

A. La existencia de desbalanceo de datos entre las distintas clases con las que se entrena un modelo supervisado no es relevante para un modelo como Naive‑Bayes.

B. Un conjunto de características como las basadas en palabras vecinas, indicando de manera binaria si aparece una palabra en un contexto o no, es suficiente para representar prácticamente cualquier texto y entrenar un modelo de ML para una tarea como la desambiguación del sentido de las palabras.

C. Si no aparecen ciertos sentidos entre las clases usadas para entrenar el modelo supervisado, no se podrán identificar cuando el texto busque desambiguar el sentido de palabras en textos nuevos.

D. Una ventaja de las características basadas en palabras vecinas es que conservan la información sobre el orden de las palabras en la oración.