1. Una vez llegados a la recta final del módulo de PLN, el alumno debe haber ejercitado ya con éxito una serie de habilidades que le permitan abordar nuevos problemas de NLP. Por ello, se propone al alumno este ejercicio, con una pequeña parte de investigación previa, orientada a una rama del NLP que aún no se ha aplicado en la asignatura: extracción de información (Information Extraction), pero que también incluye una gran cantidad de elementos que previamente han sido estudiados (NLTK, cadenas NLP, entrenamiento de taggers para el castellano, tokenización, uso de corpus de entrenamiento).

A menudo, las empresas tienen gran cantidad de información y necesitan extraer dicha información para acelerar su toma de decisiones. Dicha información suele ser pública y estar localizada en Internet en fuentes dispersas.

1. Para este ejercicio se va a imaginar que se trabaja para una empresa de envíos de comida, presente en todo el territorio nacional, con miles de pedidos cada día. Dicha empresa tiene un fichero histórico con todas las peticiones de comida que los clientes han realizado mediante el chat de su web en los últimos meses. Necesitan analizar en tiempo real qué comidas están pidiendo los usuarios y qué ingredientes tenían, ya que en la cadena de stock de alimentos es necesario realizar una previsión para no quedarse sin platos cocinados. Se ha calculado que el impacto en las ventas cada vez que uno de los platos deja de estar disponible es del 7% de pérdidas en esa semana, debido al abandono de la web de pedidos por parte del cliente. Por tanto, es de vital importancia poder realizar automáticamente estimaciones al respecto.

Como punto de partida para esta práctica, hay que recordar el último ejercicio que se hizo con WIT. WIT utiliza también la extracción de información y, tras un entrenamiento previo con las decenas o cientos de frases de ejemplo que se habían construido, era capaz de devolver la intención, la comida y los ingredientes del pedido. Se intentará algo similar con un sistema propio muy básico, para no depender de WIT. Por tanto, lo importante no es que el sistema funcione a la perfección, sino que las técnicas se apliquen adecuadamente.

1. El objetivo es programar una función que reciba como input un texto de usuario y devuelva los fragmentos de texto (chunks) que hagan referencia a las comidas y cantidades que ha solicitado. No es necesario, ni es el objetivo de este ejercicio, construir un clasificador de intención previo a esta función, sino simplemente una función que presuponemos recibe una frase con la intención 'Pedir\_comida'. Tampoco es objetivo normalizar la salida (por ej.: no es necesario convertir 'tres' a '3' ni 'pizzas' a 'pizza'). Es, por tanto, un ejercicio de mínimos.
2. Por tanto, la salida de la función será un array con diccionarios de 2 elementos (comida y cantidad). Cuando una cantidad no sea detectada, se pondrá su valor a '1' como valor por defecto.

**ANOTACIÓN**: Capítulo 7 del NLTK book

Para ello, el alumno deberá leer y asimilar por completo el capítulo 7 del NLTK book (“Extracting Information from Text”, Natural Language Processing with Python. Analyzing Text with the Natural Language Toolkit.Steven Bird, Ewan Klein, and Edward Loper, en línea <http://www.nltk.org/book/ch07.html>)1, profundizando especialmente en el apartado “3.3 Training Classifier-Based Chunkers”. En dicho capítulo se proponen distintas alternativas para la extracción de información, desde unas gramáticas similares a las CFGs, pero basadas en expresiones regulares, pasando por unigramas y bigramas y, por último, clasificadores más avanzados que usan algoritmos de clasificación como el MaxEntClassifier.

Sin embargo, el alumno deberá usar un NaiveBayesClassifier, en lugar del MaxEntClassifier, para localizar los elementos descritos anteriormente (comida y cantidad). Si el alumno no es capaz de construir un NaiveBayesClassifier —necesario para obtener un 10 en la práctica—, puede realizarlo mediante unigram o bigram tagger —para obtener un 9— o si no mediante RegexParser —un 7—.

Se deberá comenzar la práctica por el nivel más básico de dificultad (RegexParser) y, en caso de conseguirlo, añadir los siguientes niveles de forma sucesiva. De esta forma, el entregable contendrá todas y cada una de las tres formas de solucionar el problema. No basta, por tanto, con incluir, por ejemplo, únicamente un NaiveBayesClassifier, hay que incluir también las otras dos formas si se quiere obtener la máxima puntuación. Se trata simplemente de una práctica y, por tanto, no se espera como resultado un sistema de alta precisión listo para usar en producción, sino simplemente una aproximación básica que permita ejecutar las tres formas de resolver el problema.

Este ejercicio hay que hacerlo con textos de entrenamiento en español, pero teniendo en cuenta que la precisión de los POS taggers en castellano de NLTK es muy mala. Por tanto, el alumno no debe frustrarse por no obtener buenos resultados, como hemos dicho anteriormente se trata simplemente de un ejercicio teórico y podemos suponer que, con un mejor analizador, podríamos obtener mejores resultados.

Versión traducida del NLTK Book por Google Translate en este enlace: <https://translate.google.com/translate?hl=es&sl=en&tl=es&u=http://www.nltk.org/book/ch07.html>

1. Para llevar a cabo la práctica, deberá construirse una cadena NLP con NLTK, con los siguientes elementos:
   1. segmentación de frases,
   2. tokenización,
   3. POS tagger (analizador mofológico para el español).

A continuación, los POS tags obtenidos serán usados por el RegexParser, el UnigramParser, el BigramParser y el NaiveBayesClassifier.

1. Para el RegexTagger no hay que elaborar un corpus, ya que no requiere entrenamiento, sino solo definir una expresión regular adecuada. Pero para el unigram o bigram tagger, o para el NaiveBayesClassifier, lo primero que hay que hacer es construir un corpus de entrenamiento en el formato IOB, en un fichero de texto que contenga todas las frases de entrenamiento.

Este sería un ejemplo de un fragmento del fichero de entrenamiento para unigram o bigram tagger:

  yo PRP O  
  quería VBD O  
  una DT I-Cantidad  
  hamburguesa NN I-Comida

En este caso, solo se estaría entrenando con la etiqueta POS de la palabra (PRP, VBD…) como atributo (feature) para el entrenamiento.

Con NaiveBayesClassifier, teóricamente obtendremos mejores resultados y podremos incorporar información de entrenamiento adicional. El alumno deberá comprobar si esto es cierto, midiendo la precisión de cada uno de los métodos.

**El objetivo final de esta práctica es por tanto evaluar la capacidad del alumno para afrontar nuevos retos de NLP, aplicados a posibles problemas concretos de su entorno profesional. El alumno deberá demostrar su capacidad para asimilar y ejercitar nuevas técnicas y habilidades, aunque poniendo en práctica muchos de los conocimientos adquiridos anteriormente.**