Pontificia Universidad Católica del Perú

INF656, Minería Web
Trabajo práctico No. 4
Segundo semestre 2016, Master en Informática

Etiquetado gramatical de documentos

Importante: Los scripts y capturas de pantalla mostrando los resultados de los dos problemas propuestos en este TP, deben ser colocados en el repositorio creado en Paideia al terminar la clase. Si su respuesta involucra 2 o más archivos, comprímalos en un solo archivo ZIP y envíelo con su nombre. El tiempo estimado para solucionar este trabajo práctico es de 2 horas.

1. Stemming y Lemmatisation en NLTK

NLTK no solo permite hacer segmentación de palabras (como habíamos visto en el TP anterior), sino que tiene una vasta cantidad de herramientas de tratamiento de documentos textuales. Por ejemplo, gracias a NLTK podemos extraer la raíz y/o el lema de una palabra. Para ello, primero necesitamos instalar la base de datos lexical *WordNet*, que generalmente no se encuentra instalada por defecto. Esta herramienta puede instalarse desde la línea de comandos de Python, mediante los comandos siguientes:

```
>>> import nltk
>>> nltk.download('wordnet')
```

Comenzamos esta parte del trabajo práctico creando un script en Python llamado "lemma.py", que contendrá las líneas siguientes:

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
stemmer0 = PorterStemmer()
lemmatiser = WordNetLemmatizer()
print stemmer.stem("studying")
print lemmatiser.lemmatize("studying")
```

En el código anterior, la primera línea importa las dos extensiones de NLTK: *PorterStemmer* y *WordNetLemmatizer*. El primero utiliza el algoritmo de Porter de Stemming (`http://snowball.tartarus.org/algorithms/porter/stemmer.html`), mientras que el segundo, utiliza la base de datos lexical *WordNet* para realizar la tarea de lematización (`http://wordnet.princeton.edu`). Las dos lineas siguientes "instancian" las dos funciones (Stemmer y Lemmatizer) en dos variables diferentes. Finalmente, las últimas dos líneas utilizan estos métodos sobre la palabra *study* en sus diferentes flexiones.

Lance el script creado mediante el comando `python lemma.py`, analice los resultados y responda a las siguientes preguntas:

- Explique brevemente el objetivo de los dos métodos utilizados anteriormente (*stemmer* y *lemmatizer*).
- ¿Los resultados corresponden a los objetivos de los métodos?
- Intente aplicar la función *stemmer.stem* sobre la palabra "pies" ("feet" en inglés). ¿Qué obtiene como resultado? ¿Es correcto? ¿Qué pasa si prueba con la palabra "foots"?

- Ahora, utilize el método *lemmatiser.lemmatize("feet")*. ¿Está de acuerdo con el resultado obtenido?
- Por otro lado, intente aplicar la función *stemmer.stem* sobre la palabra "car's". ¿Qué obtiene como resultado?
- Existen otros métodos de stemmer implementados en NLTL, por ejemplo, el propuesto por la universidad de Lancaster. Puede utilizarlo agregando la línea `from nltk.stem.lancaster import LancasterStemmer`. "Instancie" el método con el comando `stemmer1 = LancasterStemmer()` y utilícelo sobre la palabra "car's". ¿Qué obtiene? ¿Es mejor que el método anterior?
- Ahora, pruebe con el método de stemmer llamado SnowBall. Para ello, utilice la línea `from nltk.stem import SnowballStemmer` para llamar la extensión y `stemmer2 = SnowballStemmer(''english'')` para instanciarlo. Utilice este método sobre el texto "car's". ¿Qué opina de los resultados?
- ¿Qué pasa si utiliza *lemmatiser.lemmatize("are")*? ¿Y si utiliza *lemmatiser.lemmatize("are", pos='v')*?
- ¿Por qué cree que la función *lemmatizer* utilizada sobre la misma palabra muestra dos resultados diferentes?
- Después de haber realizado todas estas pruebas, analice los resultados obtenidos y dé su opinión acerca de los dos métodos utilizados. ¿Sigue pensando que su respuesta a la primera pregunta de este bloque es correcta?

2. Part-of-Speech

En esta segunda parte, utilizaremos diferentes herramientas para realizar la tarea del etiquetado gramatical de un documento. En lingüística computacional, el etiquetado gramatical (conocido también por su nombre en inglés, part-of-speech tagging, POS tagging o POST) es el proceso de asignar (o etiquetar) a cada una de las palabras de un texto su categoría gramatical. Este proceso se puede realizar de acuerdo con la definición de la palabra o el contexto en que aparece, por ejemplo su relación con las palabras adyacentes en una frase, oración, o en un párrafo[1].

2.1. Part-of-Speech en NLTK

NLTK también cuenta con herramientas para realizar el etiquetado gramatical (PoS). Para realizar esta tarea en Python y NLTK, primero necesitamos instalar la herramienta *punkt*, que, al igual que las herramientas anteriores, no se encuentra instalada por defecto. Esta herramienta puede instalarse desde la línea de comandos de Python, mediante las líneas siguientes:

```
>>> import nltk
>>> nltk.download('punkt')
```

Comenzamos esta parte del trabajo práctico creando un script en Python llamado "pos.py", el cual, debe ser ejecutado sobre un el archivo "incognito.txt" que se encuentra en Paideia.

```
from nltk.tokenize import sent_tokenize
import sys
f = open(sys.argv[1], 'rU')
corpus = f.read()
sentences = sent_tokenize(corpus)
print sentences
```

Responda a la siguientes preguntas:

[1] http://es.wikipedia.org/

- ¿Puede identificar el contenido del texto dentro del archivo "incógnito.txt"? ¿A qué corresponde?
- ¿Qué permite hacer el script anterior?
- ¿Encontró algún problema? ¿Cuál es y en qué sentencia? ¿Que propone para solucionarlo? (no necesita implementar el código de su proposición)
- ¿Los resultados de este script son adecuados para realizar la tarea de PoS? ¿Por qué?
- Agregue una línea al código anterior para mostrar el número de elementos del vector *sentences*

Ahora, necesitamos hacer una segementación por palabras. En este sentido, podemos utilizar le código hecho en la clase anterior o utilizar el modificar el código anterior y agregar las líneas siguientes.

```
tokens = word_tokenize(corpus)
print tokens
```

Una vez el texto expresado en palabras, podemos realizar el proceso de etiquetado gramatical. Para ello, necesitamos instalar la herramienta *maxent_treebank_pos_tagger*, que generalmente no se encuentra instalada por defecto. Esta herramienta puede instalarse desde la línea de comandos de Python, mediante los comandos siguientes:

```
>>> import nltk
>>> nltk.download('maxent_treebank_pos_tagger')
```

Posteriormente, utilizamos el método *pos_tag* sobre los *tokens* extraídos en la fase anterior. Para ello, agregue el código siguiente:

```
from nltk.tag import pos_tag
tagged_tokens = pos_tag(tokens)
print tagged_tokens
```

Importante: NLTK utiliza el conjunto de etiquetas gramaticales propuestas en Peen Treebank Project (`https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`)

- Analice el resultado obtenido y discuta sobre él haciendo hincapié en los errores cometidos y en aquellos valores correctamente obtenidos.
- Lea el manual en linea de la extensión *nltk.tag* (`http://www.nltk.org/api/nltk.tag.html#module-nltk.tag`, `http://www.nltk.org/book/ch05.html`, u otro manual) y trate de mejorar los resultados obtenidos hasta el momento.
- ¿Cómo utilizaría la técnica de PoS para realizar la tarea de limpieza de los stopwords? Haga un pseudo-código o un pequeño script en Python que muestre su idea.

2.2. Etiquetador de Brill

Existen otras herramientas para realizar el etiquetado gramatical, como por ejemplo, el etiquetador de Brill, creado por Eric Brill en 1995, durante su tesis doctoral. Existe mucha información sobre esta herramienta, incluyendo, el código original (`http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/parsing/taggers/brill/0.html`).

Se sugiere a los alumnos, descargar, estudiar y lanzar experimentos utilizando este etiquetador para compararlo con los resultados obtenidos en este TP.