Datos, análisis y aplicación

Carlos Muñoz Pérez

Introducción al Procesamiento del Lenguaje Natural con Python e IA

Tercer encuentro

Introducción

En este encuentro, veremos una puesta en práctica integral de los conocimientos básicos de *Python* que adquirieron en los días previos.

- ➥ Exploraremos un proyecto que llevé a cabo hace un par de años y que tiene un cierto grado de aplicabilidad en el mundo real.

Esto nos servirá de excusa para abordar tres puntos importantes en el proceso de desarrollo de herramientas de *aprendizaje automático*.

1. La obtención de *datos* para entrenar nuestros modelos
2. El *análisis* de dichos datos (que puede ser relevante especialmente en contextos académicos)
3. El *entrenamiento* de un modelo y su *implementación* para que pueda realizar inferencias en la práctica

Escenario:

> *Son miembros del comité editorial de una revista académica de lingüística. Les llega un nuevo manuscrito acerca de algún tema oscuro de gramática. ¿Cómo elegirían a los revisores para ese artículo? ¿Contactarían a algún conocido? ¿Buscarían en Google? ¿Se fijarían quién revisó artículos similares para la revista en el pasado? Cualquiera sea la respuesta, descubrirán que encontrar revisores adecuados es un proceso que toma tiempo.*

LingPeer

Teniendo este tipo de escenario en mente, desarrollé *LingPeer*.

- ✔ *LingPeer* es una aplicación que sugiere revisores para manuscritos en lingüística teórica.
- ✔ Utiliza datos de *Lingbuzz*, un repositorio en línea de manuscritos en lingüística, para identificar expertos relevantes de forma automática.
- ✔ Está disponible como aplicación web; solo se necesitan (i) el título del artículo, (ii) sus palabras clave y (iii) el resumen.

Veamos cómo funciona *LingPeer*.

Extracción de datos

Los datos a partir de los cuales se entrenó *LingPeer* se tomaron de *Lingbuzz*.

- *Lingbuzz* es un repositorio en línea de acceso abierto que se especializa en trabajos académicos de lingüística teórica.
- Permite a los investigadores subir manuscritos, artículos, y libros, incluso si aún no se han publicado.
- Aunque no tiene el status de una revista especializada, LingBuzz ha ganado prestigio como archivo y fuente de consulta dentro de la comunidad lingüística.
- La plataforma es gestionada por Michal Starke.

Extracción de datos

Visitemos *Lingbuzz* y analicemos brevemente su código.

Extracción de datos

El *web scraping* es una técnica para extraer información de sitios web de forma automática.

- ✔ Se usan scripts o u otras herramientas que simulan la navegación web y recuperan contenido de páginas HTML.

La extracción de datos suele seguir los siguientes pasos.

1. Se realiza una solicitud a la página web.
2. Se recibe como respuesta la información contenida en la página web.
3. El programa (*scraper*) analiza la estructura de la página web.
4. Se extraen y guardan los datos relevantes (texto, imágenes, links, etc.).

Extracción de datos

El *scraping* debe hacerse de manera responsable y ética. Por lo general, se dice que hay que seguir una cierta "etiqueta".

- ✔ Las páginas tienen tienen un archivo llamado `robots.txt` que especifica qué tipo de scraping se puede realizar.
- ✔ Hay que evitar sobrecargar los servidores: hay que espaciar la frecuencia de las solicitudes, e.g., 3 segundos entre consultas.
- ✔ En mi caso, pedí permiso para tomar los datos de *Lingbuzz*, e incorporé un mecanismo de delay en el *scraper*.

Si no respetan estas condiciones, pueden banear su IP del sitio web, e incluso podría haber consecuencias legales.

Extracción de datos

Veamos algunos aspectos básicos de *web scraping*.

➥ *Notebook 1: web scraping con Python*

Exploración de los datos

Antes de aplicar modelos predictivos o de aprendizaje automático, se recomienda realizar un *análisis exploratorio de los datos* (EDA).

- ✔ El objetivo general del EDA es familiarizarse con los datos a utilizar.
- ✔ Esto involucra detectar tendencias en los datos, evaluar su calidad y formular generalizaciones sobre su comportamiento.

Vamos a hacer un breve análisis de la información obtenida a partir de *scraping*.

- ➥ Además, vamos a usar la oportunidad como excusa para mostrar algunas herramientas de Python para el análisis ed datos.

Exploración de los datos

➥ *Notebook 2: exploración de datos con Python*

Multinomial naive Bayes classifier

LingPeer se basa en un ensamble de dos clasificadores *naive Bayes*.

- Cuando el programa sugiere un revisor, lo que en realidad está haciendo es intentar predecir el autor del abstract.
- Lo que hice fue tomar las predicciones de dos modelos entrenados con diferentes reordenamientos de los datos y seleccionar los potenciales autores a partir de un cierto margen de certeza.

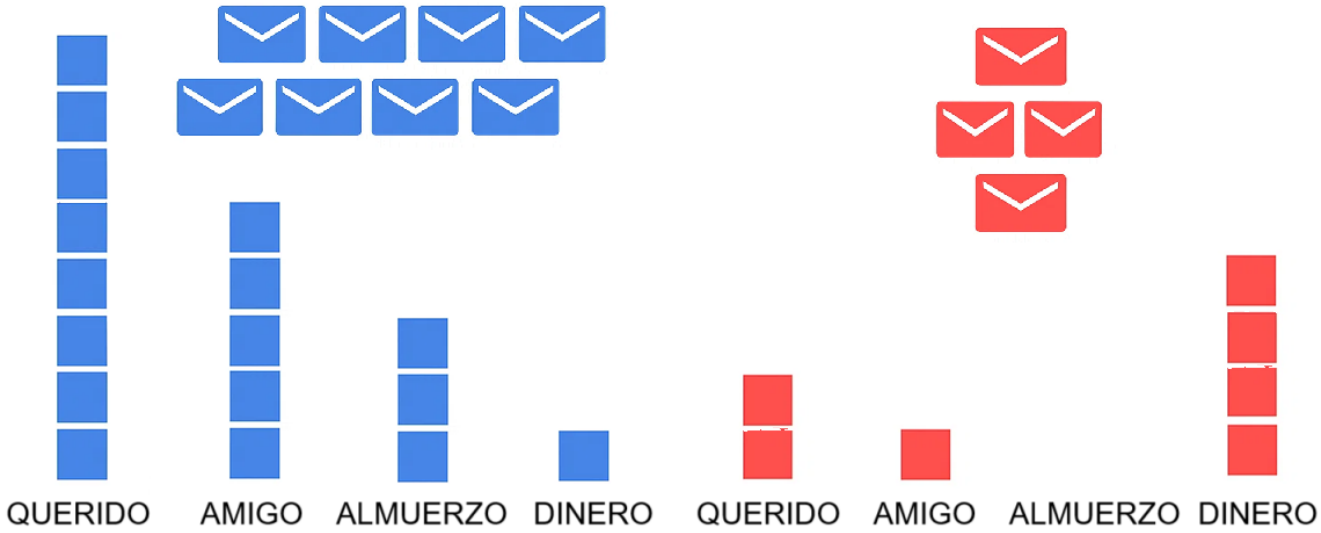
¿Es esta una metodología adecuada?

– *No sé, pero más o menos funciona.*

Multinomial naive Bayes classifier

Multinomial naive Bayes classifier

Supongan que tenemos los siguientes datos (ejemplo de *Statquest*).



Esto permite predecir si el correo *'querido amigo'* es *deseado* o *spam*.

$$8/12 \times 8/17 \times 5/17 = 0.67 \times 0.47 \times 0.29 = 0.09$$

$$4/12 \times 2/7 \times 1/7 = 0.33 \times 0.29 \times 0.14 = 0.01$$

Resultado: *querido amigo* no es spam porque $0.09 > 0.01$.

Preprocesamiento

- *Unificación del texto:* se combinan el *título* y el *resumen* (abstract) en un solo campo.
- *Lematización:* se reduce cada palabra a su forma "de diccionario" para facilitar la detección de similitudes semánticas.
- *Balance de etiquetas:* se normaliza el conjunto de entrenamiento para que haya una cantidad similar de ejemplos por etiqueta.
- *Vectorización:*
  - Convertir el texto en un vector numérico.
  - Eliminar *stopwords*.
  - Filtrar términos muy frecuentes.
  - Convertir todo el texto a minúsculas.
- *División de datos:* separación en conjuntos de entrenamiento y prueba.

Preprocesamiento

Vayamos a ver el código que se supone hace todo eso.

➥ *Notebook 3: naive Bayes classifiers*

Preprocesamiento

Para *LingPeer* apliqué un método de preprocesamiento que se me ocurrió en el momento.

1. Lematizar **todas** las palabras clave del repositorio
2. Reemplazar palabras clave de dos o más ítems (*n-grams*) por una sola palabra, e.g., *contrastive topic* $\rightarrow$ *ngram0021*
3. Lematizar abstract
4. Reemplazar (*n-grams*) en el abstract

Streamlit

- *Streamlit* es un framework de código abierto en Python.
- Permite crear *aplicaciones web* de forma rápida y sencilla.
- Sintaxis *simple* y orientada a scripts.
- No se necesitan conocimientos de HTML, CSS o JavaScript.
- Integración nativa con bibliotecas como Pandas y Matplotlib.
- Actualización automática al modificar el código en *GitHub*.

Un breve ejemplo del código que usa *Streamlit*.

```
st.title('LingPeer')

st.markdown("<i>Get potential reviewers for papers in
    theoretical linguistics based on data from [Lingbuzz](
    https://ling.auf.net)</i>", unsafe_allow_html=True)

title = st.text_input("Title")

keywords = st.text_input("Keywords", help='For better
    results, introduce at least 3 keywords separated by
    commas or semicolons')

abstract = st.text_area("Abstract", height=50, help='The
    abstract must have a maximum lenght of 1000 words')
```

Streamlit

```
if st.button("Suggest me reviewers!"):
    peers = get_peers(title, keywords, abstract)
    ...
        for name, kw_list, title, ms_id, _ in peers:
            ...
            if len(kw_list) == 0:
                st.write('No keywords in common between this
                    author and the info you provided.')
            else:
                st.write('This author has employed the
                    following matching keywords.')
                kw_acum = []
                ...
                kw_acum = '\n'.join(kw_acum)
                st.markdown(kw_acum)

            url = f'https://ling.auf.net/{ms_id}'
            st.markdown(f'As a reference, you can check
                their manuscript *[{title}]({url})*.')
```

Código

Pueden revisar el código que vimos en esta presentación en mi página de GitHub.

- ✔ *lingbuzz_scraper*
- ✔ *lingbuzz_data_analysis*
- ✔ *lingbuzz_abstract_classifier*
- ✔ *LingPeer*

No lo usen como código de referencia. Me agarró un poco de vergüenza volver a ver este código.

Algunas recomendaciones

- *Foundations of Python Programming:* Libro-curso online de Python, con ejercicios.
- *Statquest:* recurso educativo, principalmente un canal de YouTube y sitio web, creado por Josh Starmer, diseñado para explicar conceptos de estadística, ciencia de datos y machine learning de forma clara y accesible
- *Kaggle:* una plataforma en línea para ciencia de datos y machine learning, con cursos y competencias