Preprocesamiento de datos.

David Eduardo Gallardo Fernández
Monterrey, N.L., México
Email: david.gallardofdz@uanl.edu.mx

***Abstract*—Con el avance en el poder de procesamiento que han tenido las computadoras y el abaratamiento en los costos de almacenamiento de datos en los últimos años se ha hecho posible desarrollar lo que conocemos como Big Data. Ahora es posible capturar, almacenar, procesar y analizar grandes volúmenes de información para obtener datos que den un valor y permitan tomar decisiones mas acertadas. El análisis de sentimiento permite analizar lo que un texto trata de decir y si lo que dice es positivo, negativo o neutral. Antes de poder realizar un análisis de sentimiento es necesario pre-procesar los datos o el texto a fin de poder realizar un análisis mas acertado. En este reporte se explicarán algunas técnicas de pre-procesamiento de texto que se pueden utilizar previo al análisis de sentimiento.**

Keywords: text, sentiment analysis.

I. Introducción

El pre-procesamiento de datos es el o los pasos previos durante el proceso de limpieza de datos. Comprende cualquier tipo de procesamiento que se realiza con los datos brutos para transformarlos en datos que tengan formatos que sean más fáciles de utilizar o para quitar todos aquellos datos que no son necesarios o que no aportan ningún valor. En este reporte se analizan las siguientes actividades:

- Convertir todo el texto a minúsculas.
- Quitar los números.
- Quitar signos de puntuación.
- Quitar stopwords.
- Stemming.
- Lemmatizer

II. Fuente de datos

Para el presente trabajo se obtuvieron los datos de wikipedia utilizando la libreria wikipedia para python. El tema que se buscó fué el de ”Genealogical DNA test” [1]. El código fuente está en Github [4].

III. Procedimiento

A. Obtener los datos

Para obtener los datos instalamos la librería wikipedia y luego hacemos referencia a ella para poder utilizarla. Haciendo uso de la función ”wiki.page” traemos la información de la página que nos interesa procesar.

```
wiki_topic = wiki.page('Genealogical DNA test')
```

Listing 1: Python example

B. Convertir todo el texto a minúsculas.

El convertir todo el texto a minúsculas es un paso clave y prácticamente estándar en todo proceso de limpieza de texto ya que nos permite identificar las palabras únicas en el texto, de otro modo ”prueba”, ”Prueba” y ”PRUEBA” serían tres palabras diferentes para el algoritmo y esto dificultaría el análisis.

C. Quitar los números.

Los números generalmente no forman parte de las palabras y no nos ayudan a identificar el tema central ni el sentimiento del texto por lo que es necesario removerlos del texto. Esto se puede llevar a cabo usando expresiones regulares como se muestra a continuación.

```
re.sub(r'\d+', '', text)
```

Listing 2: Quitar los números.

D. Quitar signos de puntuación.

Los signos de puntuación al igual que los números generalmente no forman parte de las palabras y no nos ayudan a identificar el tema central ni el sentimiento del texto por lo que es necesario removerlos del texto. En este caso lo llevamos a cabo como se muestra a continuación.

```
text.translate(str.maketrans(' ', ' ', string.punctuation))
```

Listing 3: Quitar los signos de puntuación.

E. Quitar stopwords.

Las ”stopwords” son palabras que aparecen con mucha frecuencia en los textos pero son vacías o no tienen sentido cuando se escriben solas y no ayudan a entender de que se trata el texto que se está analizando. Al ser palabras vacías lo mejor es quitarlas para reducir la cantidad de información que se tienen que procesar y así reducir el costo computacional necesario para analizar el texto. Generalmente las stopwords son básicamente, conjunciones, artículos, preposiciones y adverbios.

```
def remove_stopwords(text):
    stop_words = set(stopwords.words("english"))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    return filtered_text
```

Listing 4: Funció para quitar las stopwords.

Para este trabajo se utilizó la libreria NLTK con el idioma inglés ya que el texto esta en inglés.

F. Stemming.

Stemming es el proceso que consiste en reducir una palabra a su raíz o, en inglés, a un stem. Por ejemplo: zapato, zapatero y zapatería tienen la misma raíz que es zapat entonces al aplicar el steming todas estas palabras se convertirán a ”zapato” y de esta forma el vocabulario del texto que se esta analizando se puede reducir aun más.

La librería NLTK también tiene la capacidad de realizar el stemming y la función utilizada para llevarla a bajo se encuentra acontinuación.

```
def stem_words(text):
    word_tokens = word_tokenize(text)
    stems = [stemmer.stem(word) for word in
    word_tokens]
    return stems
```

Listing 5: Stemming usando NLTK.

G. Lemmatizer

La lematización consiste en hallar el lema correspondiente, dada una forma flexionada es decir, en plural, en femenino, conjugada, etc. El lema es la forma que se acepta como representante de todas las formas flexionadas de una misma palabra. Es decir, el lema de una palabra es la palabra que nos encontraríamos como entrada en un diccionario tradicional: singular para sustantivos, masculino singular para adjetivos, infinitivo para verbos. Por ejemplo guapo es el lema de guapas, mesa es el lema de mesas.

Lematizar implica estandarizar, desambiguar, segmentar y, en caso de usar programas de lematización automática, también etiquetar.

```
lemmatizer = WordNetLemmatizer()

def lemmatize_word(text):
    word_tokens = word_tokenize(text)
    # provide context i.e. part-of-speech
    lemmas = [lemmatizer.lemmatize(word, pos
    ='v') for word in word_tokens]
    return lemmas
```

Listing 6: Lematización del texto.

IV. Visualización de Datos

La visualización de datos facilita el entendimiento de los mismos. Para proyectos de análisis de texto como en este caso existen varias opciones, por ejemplo, podemos ver la frecuencia de cada palabra con una gráfica de línea [Fig 1] y con una nube de palabras en la cual la palabra mas grande es la que se repite en mas ocasiones [Fig 2].

V. Conclusiones

Al pre-procesar el texto podemos obtener las palabras que realmente se necesitan para llevar a cabo el análisis de sentimiento. Las librerías como ”nltk” nos ayudan a limpiar las palabras que no necesitamos estandarizar, por llamarlo de alguna manera, las palabras que realmente están relacionadas con el tema central y con las palabras que nos pueden ayudar a saber el sentimiento del texto. Este pre-procesamiento es muy importante y parte clave para poder realizar un buen análisis ya que sin este el análisis de sentimiento podría ser inexacto o requerir un mayor costo computacional.

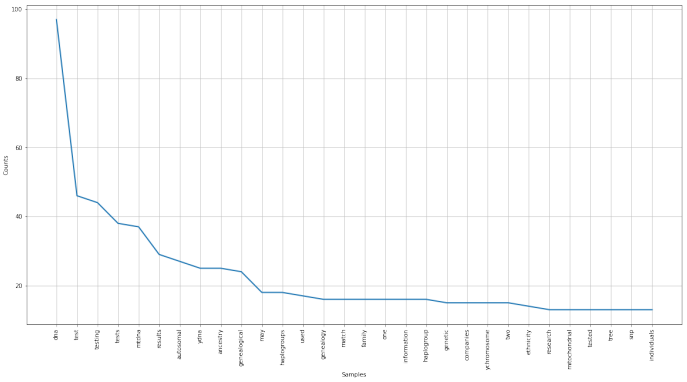

Fig. 1: Frecuencia de palabras

Fig. 2: Nube de palabras

References

[1] “Genealogical DNA test”, Wikipedia. el 6 de mayo de 2022. Consultado: el 19 de mayo de 2022. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Genealogical$_DNA_test oldid = 1086512802

[2] “Stemming and lemmatization”. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (consultado el 19 de mayo de 2022)

[3] J. Goldsmith, Wikipedia. 2022. Consultado: el 19 de mayo de 2022. [En línea]. Disponible en: https://github.com/goldsmith/Wikipedia

[4] degallardo, MCD-PCD. 2022. Consultado: el 19 de mayo de 2022. [En línea]. Disponible en: https://github.com/degallardo/MCD-PCD/