Reporte
“Universal sentence encoder”

Armenta García Guadalupe Javier

Para la realización de esta práctica se hizo uso de la implementación que tiene tensorflow de este algoritmo.

Primero se instalaron las librerias pertinentes:

- sudo pip install tensorflow
- sudo pip install tensorflow_hub
- sudo pip install matplotlib
- sudo pip install numpy

El módulo de Universal sentence encoder pertenece a google por eso se tiene que importar desde su link del repo y se carga usando tensorflow_hub

```
: from absl import logging

import tensorflow as tf
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import numpy as np

module_url = "https://tfhub.dev/google/universal-sentence-encoder/4" #@param ["https://tfhub.dev/google/universal-s
model = hub.load(module_url)
print ("module %s loaded" % module_url)
```

```
module https://tfhub.dev/google/universal-sentence-encoder/4 loaded
```

Las siguientes fueron las funciones usadas

```
def embed(input):
    return model(input)

def plot_similarity(labels, features, rotation):
    corr = np.inner(features, features)
    sns.set(font_scale=1.2)
    g = sns.heatmap(
      corr,
      xticklabels=labels,
      yticklabels=labels,
      vmin=0,
      vmax=1,
      cmap="YlOrRd")
    g.set_xticklabels(labels, rotation=rotation)
    g.set_title("Semantic Textual Similarity")

def run_and_plot(messages_,message_embeddings_ ):
    plot_similarity(messages_, message_embeddings_, 90)
```

- Embed: recibe como entrar el conjunto de documentos y se los envía al modelo importado que pesa alrededor de 1gb. (en realidad no tiene alguna utilidad más que especificar que es el “embed”
- Plot_similarity: es la función que va a calcular y graficar la similitud entre los documentos (todos contra todos) en una tabla por medio de calcular el producto interno de las features (siendo features los vectores

asociados a cada documento), además de insertar parámetros de diseño que requiere matplotlib para graficar.

- Run_and_plot: solo invoca la función anterior enviándole los documentos, los vectores generados por los algoritmos generados y la especificación de rotación (es decir el ángulo que tendrán las etiquetas en el gráfico.

```
In [20]: document_0 = 'Economic news have little effect on financial markets'
         document_1 = 'eat with wooden spoon'
         document_2 = 'eat with metallic spoon'

         messages = [document_0, document_1, document_2]

         message_embeddings = embed(messages)

         print("Embedding size: {}".format(len(message_embedding)))

         for i, message_embedding in enumerate(np.array(message_embeddings).tolist()):
             print("Message: {}".format(messages[i]))
             message_embedding_snippet = ", ".join((str(x) for x in message_embedding[:3]))
             print("Embedding: [{}, ...]\n".format(message_embedding_snippet))
```

- Se declaran los documentos, se insertan en una lista y esta lista es enviada al modelo para obtener las features “message_embeddings”.
- Se imprime el tamaño de los vectores que son en total 512 features por cada doc
- Para poder iterar sobre message_embeddings se convierte a array de tipo numpy se usa enumerate para darles un índice y se invoca el metodo to list para que sea un iterable
- Se imprime el documento, y se formatea el vector para que se muestren algunas features.

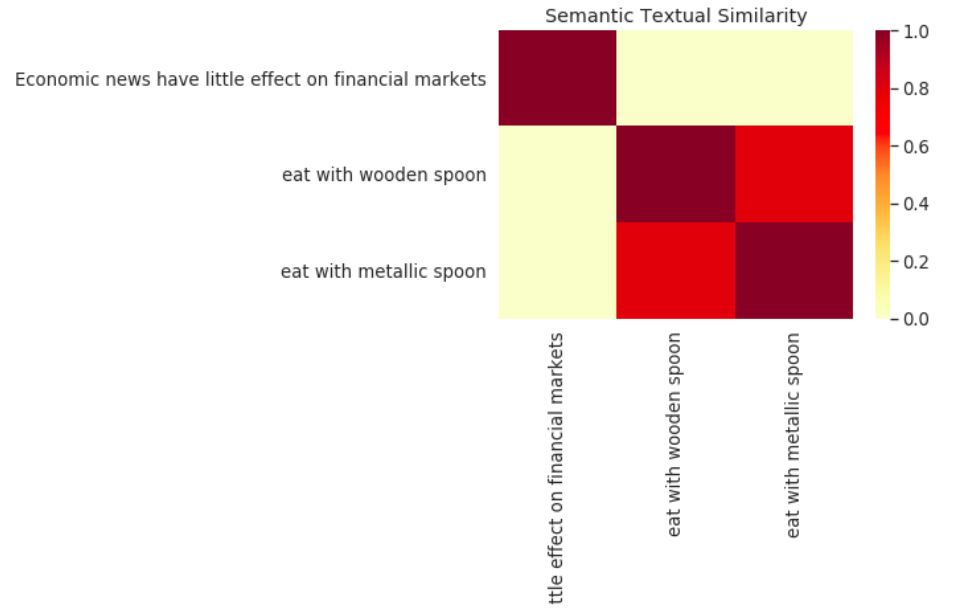
```
Embedding size: 512
Message: Economic news have little effect on financial markets
Embedding: [0.029726974666118622, 0.004525386728346348, 0.003189939772710204, ...]

Message: eat with wooden spoon
Embedding: [0.005436183884739876, 0.05711539462208748, 0.052789684385061264, ...]

Message: eat with metallic spoon
Embedding: [0.030168740078806877, 0.0445793978869915, 0.06876149773597717, ...]
```

Finalmente se manda llamar a la función para calcular la similitud y graficarla:

```
run_and_plot(messages,message_embeddings)
```



Como se puede ver los documentos semanticamente más similares son:

- document_1 = 'eat with wooden spoon'
- document_2 = 'eat with metallic spoon'