PNL - Especialización en inteligencia artificial

Memoria descriptiva de los desafíos realizados durante el cursado de la materia.

Consigna del desafío 1

1- Vectorizar documentos. Tomar 5 documentos al azar y medir similaridad con el resto de los documentos. Estudiar los 5 documentos más similares de cada uno analizar si tiene sentido. La similaridad según el contenido del texto y la etiqueta de clasificación.

2- Entrenar modelos de clasificación Naïve Bayes para maximizar el desempeño de clasificación (f1-score macro) en el conjunto de datos de test. Considerar cambiar parámetros de instanciación del vectorizador y los modelos y probar modelos de Naïve Bayes Multinomial y ComplementNB.

3- Transponer la matriz documento-término. De esa manera se obtiene una matriz término-documento que puede ser interpretada como una colección de vectorización de palabras.

Estudiar ahora similaridad entre palabras tomando 5 palabras y estudiando sus 5 más similares. ****La elección de palabras no debe ser al azar para evitar la aparición de términos poco interpretables, elegirlas "manualmente"****.

Desarrollo

Para la realización de este desafío se utilizó como base la notebook vista en clase, en donde se realiza la vectorización de texto y modelo de clasificación Naïve Bayes con el dataset 20 newsgroups.

El dataset 20 newsgroups se descarga directamente de la librería ScikitLearn.

El corpus se vectoriza utilizando TfidfVectorizer de la librería ScikitLearn.

Dicho corpus consta de:

- cantidad de documentos: 11314
- tamaño del vocabulario (dimensionalidad de los vectores): 101631
- hay 20 clases correspondientes a los 20 grupos de noticias

```
'alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc',
omp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.windows.x',
'misc.forsale', 'rec.autos', rec.motorcycles', 'rec.sport.baseball',
'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space',
'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast',
'talk.politics.misc', 'talk.religion.misc'
```

Modelo de clasificación Naïve Bayes

Se utilizan los modelos MultinomialNB y ComplementNB de la librería sklearn.naive_bayes

Punto 1.

Vectorizar documentos. Tomar 5 documentos al azar y medir similaridad con el resto de los documentos. Estudiar los 5 documentos más similares de cada uno analizar si tiene sentido. La similaridad según el contenido del texto y la etiqueta de clasificación.

Documento original N°10440 (Categoría: rec.sport.hockey):

These new rule changes are great! However, I think that your rules are MUCH too complicated. How will the normal average fan be able to count how many fouls a player has? And then we would even ha...

********** Documentos más similares: **********

DOCUMENTO SIMILAR: 1

Índice: 9434
Similitud: 0.3907

Categoría: rec.sport.baseball
Texto:

What makes you think Buck will still be in New York at year's end with George back? :-)

--
Keith Keller LET'S GO RANGERS!!!!!
LET'S GO QUAKERS!!!!!
kkeller@mail.sas.upenn.edu IVY LE...

Similaridad

- *Documento N°1:*

Tema: rec.sport.hockey

Documentos similares:
1- Índice: 9434
Similitud: 0.3907
Categoría: rec.sport.baseball
2- Índice: 10511
Similitud: 0.2546
Categoría: comp.graphics
3- Índice: 10078
Similitud: 0.2370
Categoría: rec.motorcycles
4- Índice: 7139
Similitud: 0.2290
Categoría: talk.politics.misc
5- Índice: 9858
Similitud: 0.2205
Categoría: comp.os.ms-windows.misc

- *Documento N°2:*

Tema: comp.os.ms-windows.misc

Documentos similares:
1- Índice: 346
Similitud: 0.3567
Categoría: comp.sys.ibm.pc.hardware
2- Índice: 3254
Similitud: 0.2951
Categoría: comp.graphics
3- Índice: 5971
Similitud: 0.2947
Categoría: comp.sys.ibm.pc.hardware
4- Índice: 6927
Similitud: 0.2742
Categoría: comp.sys.ibm.pc.hardware
5- Índice: 1782
Similitud: 0.2718
Categoría: comp.os.ms-windows.misc

- *Documento N°3:*

Tema: misc.forsale

1- Índice: 765
   Similitud: 0.2520
   Categoría: comp.sys.mac.hardware
2- Índice: 7613
   Similitud: 0.2418
   Categoría: comp.sys.mac.hardware
3- Índice: 1083
   Similitud: 0.2193
   Categoría: comp.os.ms-windows.misc
4- Índice: 8141
   Similitud: 0.2116
   Categoría: sci.electronics
5- Índice: 11283
   Similitud: 0.1980
   Categoría: comp.os.ms-windows.misc

- *Documento N°4:*

Tema: sci.electronics

1- Índice: 5571
   Similitud: 0.1306
   Categoría: misc.forsale
2- Índice: 428
   Similitud: 0.1086
   Categoría: comp.sys.mac.hardware
3- Índice: 3140
   Similitud: 0.1086
   Categoría: comp.sys.mac.hardware
4- Índice: 9266
   Similitud: 0.1086
   Categoría: rec.motorcycles
5- Índice: 7811
   Similitud: 0.1072
   Categoría: comp.sys.ibm.pc.hardware

- *Documento N°5:*

Tema: comp.graphics

1- Índice: 8270
   Similitud: 0.3771
   Categoría: comp.os.ms-windows.misc
2- Índice: 10511
   Similitud: 0.2546
   Categoría: comp.graphics
3- Índice: 10078
   Similitud: 0.2370

Categoría: rec.motorcycles
4- Índice: 7139
Similitud: 0.2290
Categoría: talk.politics.misc
5- Índice: 9858
Similitud: 0.2205
Categoría: comp.os.ms-windows.misc

Punto 2.

Entrenar modelos de clasificación Naïve Bayes para maximizar el desempeño de clasificación (f1-score macro) en el conjunto de datos de test. Considerar cambiar parámetros de instanciación del vectorizador y los modelos y probar modelos de Naïve Bayes Multinomial y ComplementNB.

Se entrenan y evalúan modelos de clasificación de texto, utilizando dos algoritmos de Naive Bayes (MultinomialNB y ComplementNB), optimizando los hiperparámetros mediante búsqueda en cuadrícula (GridSearchCV), y analizando errores de clasificación.

El resultado es este:

```
Entrenando MultinomialNB...
Fitting 5 folds for each of 54 candidates, totalling 270 fits

Mejores parámetros para MultinomialNB:
{'vectorizer__max_df': 0.8, 'vectorizer__max_features': 10000, 'vectorizer__min_df': 5,
'vectorizer__ngram_range': (1, 1)}

F1-score (macro) en validación cruzada: 0.6533
F1-score (macro) en test: 0.6182

Reporte de clasificación detallado:
              precision    recall  f1-score   support

           0       0.67      0.15      0.25       319
           1       0.64      0.67      0.66       389
           2       0.66      0.55      0.60       394
           3       0.57      0.72      0.64       392
           4       0.77      0.61      0.68       385
           5       0.79      0.75      0.77       395
           6       0.82      0.74      0.78       390
           7       0.76      0.70      0.73       396
           8       0.81      0.73      0.77       398
           9       0.90      0.76      0.82       397
          10       0.58      0.89      0.70       399
```

```
      11      0.65      0.75      0.69       396
...
Top 3 probabilidades:
- soc.religion.christian: 0.0547
- rec.autos: 0.0546
- talk.politics.misc: 0.0522
```

Resumen:

Modelo: MultinomialNB

Mejores parámetros
'vectorizer__max_df': 0.8, 'vectorizer__max_features': 10000, 'vectorizer__min_df': 5, 'vectorizer__ngram_range': (1, 1)}

Iterpretación:
'vectorizer__max_df': 0.8 ==> Se eliminan las palabras muy frecuentes. Debe ser porque si se repiten tanto, no deben ser tan importantes
'vectorizer__min_df': 5 ==> Se eliminan las palabras que tienen poca repitencia.
'vectorizer__max_features': 10000 ==> Prueba con las 10000 más frecuentes
'vectorizer__ngram_range': (1, 1) ==> La mejor opción es con unigramas. Parece que permite desmenusar mejor las palabras

F1-score (macro) en validación cruzada: 0.6533
F1-score (macro) en test: 0.6182

Conclusión:
El modelo tiene algo de overfiting, ya que la métrica de validación tiene un valor inferior a la test.

Modelo: ComplementNB

Mejores parámetros
{'vectorizer__max_df': 0.8, 'vectorizer__max_features': 50000, 'vectorizer__min_df': 1, 'vectorizer__ngram_range': (1, 1)}

Iterpretación:
'vectorizer__max_df': 0.8 ==> Se eliminan las palabras muy frecuentes. Debe ser porque si se repiten tanto, no deben ser tan importantes
'vectorizer__min_df': 1 ==> Se eliminan las palabras que tienen poca repitencia. En este caso es mas estricto que el modelo MultinomialNB
'vectorizer__max_features': 50000 ==> Prueba con las 50000 mas frecuentes
'vectorizer__ngram_range': (1, 1) ==> La mejor opcion es con unigramas. Parece que permite desmenusar mejor las palabras

F1-score (macro) en validación cruzada: 0.7535
F1-score (macro) en test: 0.6933

Conclusión:
El modelo performa mejor que MultinomialNB, pero se observa también overfiting.

Punto 3

> Transponer la matriz documento-término. De esa manera se obtiene una matriz término-documento que puede ser interpretada como una colección de vectorización de palabras.

Resultados:

Las palabras elegidas fueron: *science, tree, clothes, vacation, electronics*

Palabra: science
Las 5 más similares:
- scientific: 0.3336
- empirical: 0.2737
- hypotheses: 0.1890
- fiction: 0.1842
- scientists: 0.1781

Observaciones: Todas las palabras tienen alta relación con la palabra buscada

Palabra: tree
Las 5 más similares:
- immaculate: 0.2701
- righteous: 0.2575
- palm: 0.2472
- u2: 0.2131
- christmas: 0.2078

Observaciones: Acá se ve confusión.
immaculate - No sé qué relación encuentra
righteous - No sé qué relación encuentra
palm - Palmera, es una variedad de árbol
u2 - Supongo que la banda musical. No sé qué relación encuentra
christmas - Relacionada por árbol de navidad

Palabra: clothes
Las 5 más similares:
- whites: 0.3479
- cramer: 0.2502
- coat: 0.2463
- clothing: 0.2445
- substitute: 0.2373

Observaciones: Algunas bien, algunas mal
whites - (BLancas) Tiene relación con la palabra ropa
cramer - No sé qué significa
coat - Campera/abrigo, tiene relación
clothing - Sinónimo de ropa
substitute - No sé qué relación encuentra

Palabra: vacation
Las 5 más similares:
- las: 0.3774
- orlando: 0.2984
- vegas: 0.2732
- hotel: 0.2666
- trip: 0.2333

Observaciones: Algunas bien, algunas mal
las: Supongo que lo relaciona con Las Vegas, está bien, pero corta la palabra
orlando: Correcto
vegas: Supongo que lo relaciona con Las Vegas, está bien, pero corta la palabra
hotel: Correcto
trip: Correcto

Palabra: electronics
Las 5 más similares:
- 805: 0.1738
- addressed: 0.1646
- diagram: 0.1627
- mq: 0.1520
- females: 0.1477

Observaciones: Algunas bien, algunas mal

805: No sé qué relación encuentra
addressed: Tiene relación
diagram: Tiene relación
mq: No sé qué significa "mq"
females: No sé qué relación encuentra

Consigna del desafío 2

- Crear sus propios vectores con Gensim basado en lo visto en clase con otro dataset.

- Probar términos de interés y explicar similitudes en el espacio de embeddings (sacar conclusiones entre palabras similitudes y diferencias).

- Graficarlos.

- Obtener conclusiones.

Desarrollo

El conjunto de datos a utilizar es el correspondiente a la base de datos "Enron Email Dataset". Contiene datos de alrededor de 150 usuarios con un total de alrededor de 500,000 mensajes de correo electronico. Originalmente el dataset se hizo público por la Comision Federal Regulatoria de Energia de Estaos Unidos durante la investigacion alrededor del colapso de la empresa Enron.

Instancia del modelo

```
# Crearmos el modelo generador de vectores
# En este caso utilizaremos la estructura modelo Skipgram
w2v_model = Word2Vec(min_count=10,    # frecuencia mínima de palabra para incluirla en el vocabulario
                     window=2,       # cant de palabras antes y desp de la predicha
                     vector_size=300,       # dimensionalidad de los vectores
                     negative=20,    # cantidad de negative samples... 0 es no se usa
                     workers=1,      # si tienen más cores pueden cambiar este valor
                     sg=1)           # modelo 0:CBOW  1:skipgram
```

Cantidad de documentos en el corpus: 5854

Entrenamiento

```
# Entrenamos el modelo generador de vectores
# Utilizamos nuestro callback
w2v_model.train(sentence_tokens,
                total_examples=w2v_model.corpus_count,
                epochs=20,
                compute_loss = True,
                callbacks=[callback()]
                )
```

Análisis de las palabras similares para cada ejemplo:

Como es un dataset de correos Spam, se escogieron palabas que supongo pueden ser frecuentes en el cuerpo o asunto del correo:

1 - "congratulations": 3/5
('congrats', 0.6032657623291016) : correcto
('deserved', 0.49680399894714355): (Merecido) correcto
('wolfin', 0.38669973611831665) : No reconozco la palabra
('merry', 0.37556132674217224) : (Feliz) correcto
('macmillan', 0.3686480224132538 : No reconozco la palabra

2 - "promotion": 4/5
('congrats', 0.4194207787513733) : correcto
('deserved', 0.3881804049015045) : (merecido), puede que un contexto dado sea correcto
('deserving', 0.3827337622642517): (merecedor), puede que un contexto dado sea correcto
('election', 0.38179779052734375): (elección - elegido), puede que un contexto dado sea correcto
('shannon', 0.358280211687088) : No reconozco la palabra

3 - "cards": 2/5
('debit', 0.5138387084007263) : correcto
('acts', 0.4018385112285614) : incorrecto
('depot', 0.3781992495059967) : correcto
('payroll', 0.3724546432495117) : correcto
('stored', 0.36612412333488464) : incorrecto

4 - "purchasing": 0/5
('uhc', 0.35434094071388245) : No reconozco la palabra, incorrecto
('lan', 0.3450513780117035) : incorrecto
('decrease', 0.34434276819229126) : incorrecto
('coulter', 0.33327993750572205) : incorrecto
('consolidated', 0.3308396637439728) : incorrecto

5 - "new": 1/5
('york', 0.40912848711013794) : Lo asocia a la ciudad, correcto
('healthy', 0.3450511693954468) : incorrecto
('lap', 0.32198408246040344) : incorrecto
('vdg', 0.3171232044696808) : incorrecto
('fort', 0.31698963046073914) : incorrecto

Consigna del desafío 3

Utilizar otro dataset y poner en práctica la generación de secuencias con las estrategias presentadas.

Desarrollo

Para el desarrollo de este desafío cambie el corpus y coloque el texto de La Biblia, no poque sea religioso, se me ocurrio por ser un libro disponible en la red y de gran extensión.

Y me encontré que rápidamente sobrepasaba la RAM de Colab. Por eso verá que me quede con la décima parte del texto.

Asimismo tambien tuve muchos problemas con la RAM, al terminar la época 1 se desbordaba, por eso reduje el batch size.

Con T4 cada época demora unos 6 minutos, con CPU demoran 25min, y es bastante complicado hacer pruebas variando parámetros con esos tiempos.

La longitud del corpus es: 555956 caractéres.

Verá estos cambios en la red:

- Cambie la capa SimpleRNN a una LSTM. Le puse 150 neuronas.

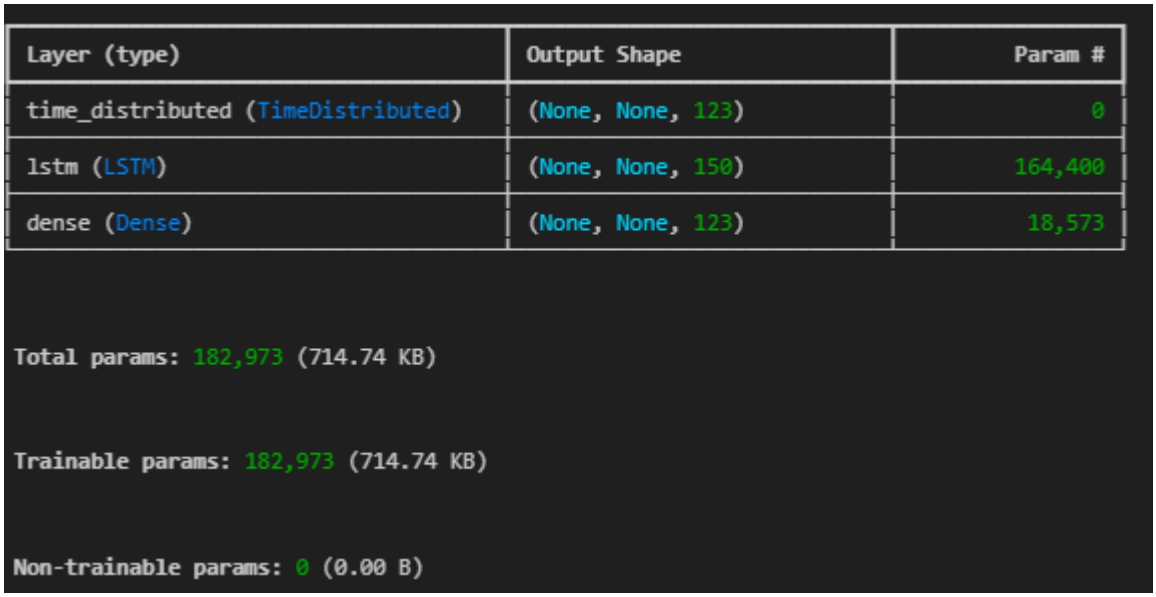
- En esta capa, cambie los dropout desde 0.1 a 0.2

- Cambie el optimizador a Adam

- Fije el max_context_size = 80

- Deje en entrenamiento en 10 épocas, ya que sino demoraba mucho y no podria hacer cambios

- Modifique la fucnión: def on_epoch_end(self, epoch, logs=None) para que procese los datos por batch



Layer (type)	Output Shape	Param #
time_distributed (TimeDistributed)	(None, None, 123)	0
lstm (LSTM)	(None, None, 150)	164,400
dense (Dense)	(None, None, 123)	18,573

Total params: 182,973 (714.74 KB)

Trainable params: 182,973 (714.74 KB)

Non-trainable params: 0 (0.00 B)

Generación de secuencias

input_text= 'Las personas'

output= 'Las personas de la tierra de la tier'

Consigna del desafío 4

Construir QA Bot basado en el ejemplo del traductor, pero con un dataset QA

Desarrollo

Para este desafío trabaje con el notebook base de bot_qa.

Hice algunos cambios pequeños en la matriz de embedding para subsanar algunos errores. Agregue una función callback para el earlystopping.

Model: "functional_9"

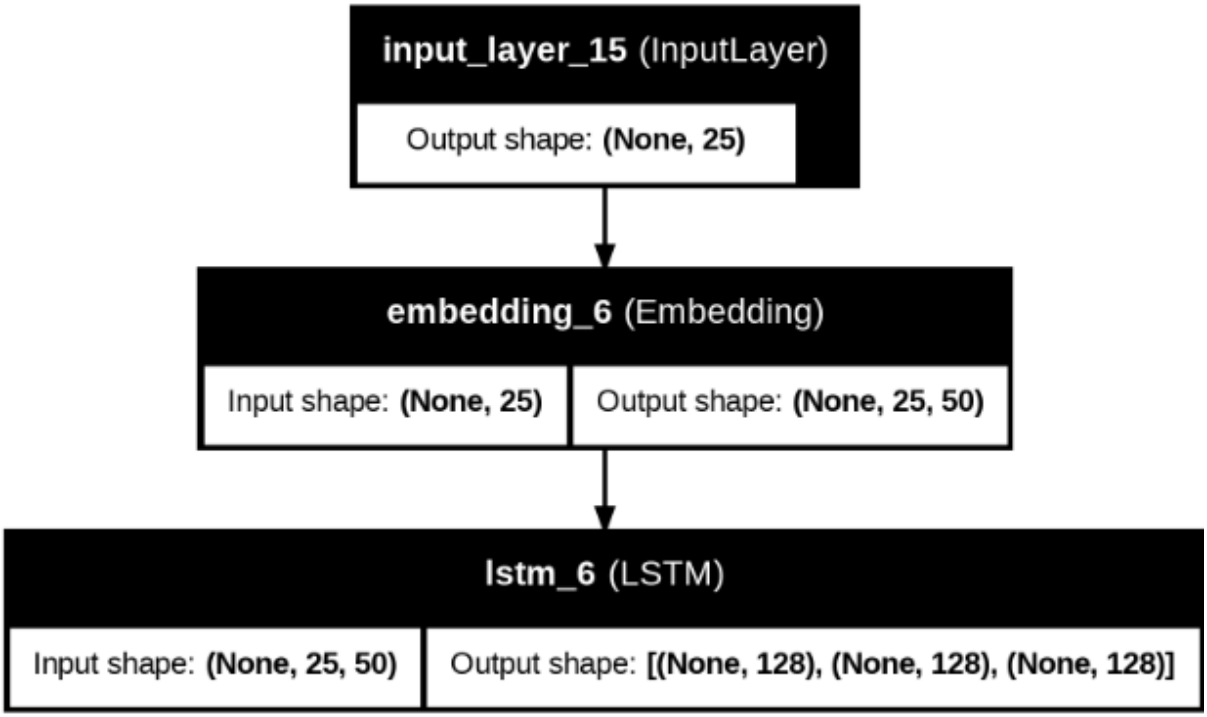
Layer (type)	Output Shape	Param #	Connected to
input_layer_15 (InputLayer)	(None, 25)	0	-
input_layer_16 (InputLayer)	(None, 10)	0	-
embedding_6 (Embedding)	(None, 25, 50)	90,000	input_layer_15[0][0]
embedding_7 (Embedding)	(None, 10, 128)	231,552	input_layer_16[0][0]
lstm_6 (LSTM)	[(None, 128), (None, 128), (None, 128)]	91,648	embedding_6[0][0]
lstm_7 (LSTM)	[(None, 10, 128), (None, 128), (None, 128)]	131,584	embedding_7[0][0], lstm_6[0][1], lstm_6[0][2]
dense_3 (Dense)	(None, 10, 1809)	233,361	lstm_7[0][0]

Total params: 778,145 (2.97 MB)

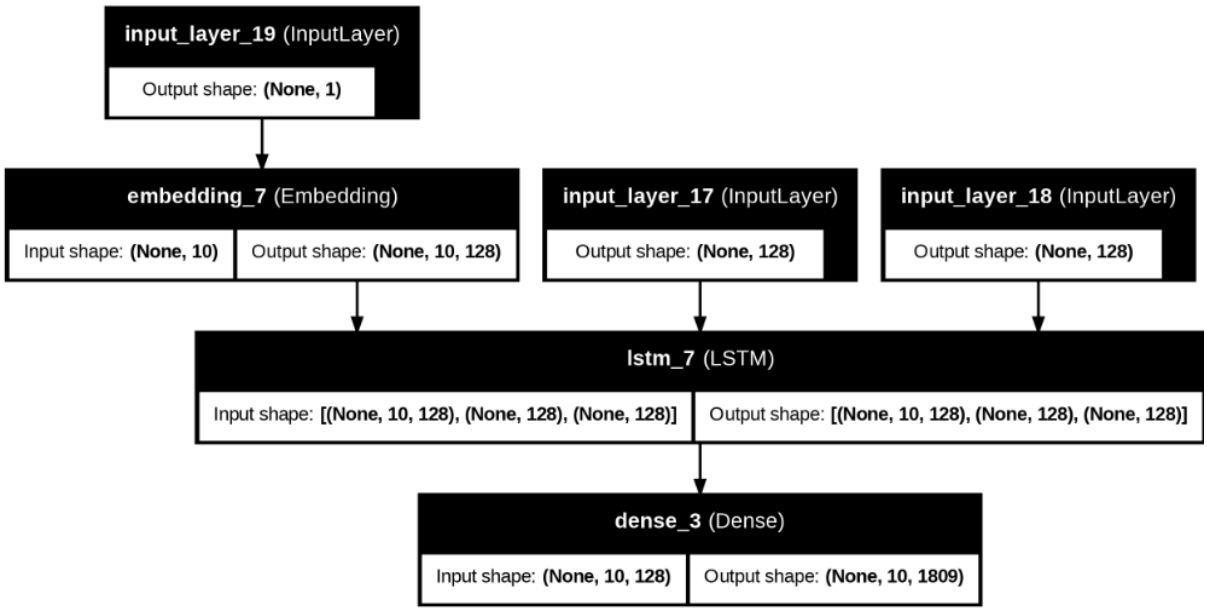
Trainable params: 688,145 (2.63 MB)

Non-trainable params: 90,000 (351.56 KB)

Encoder:



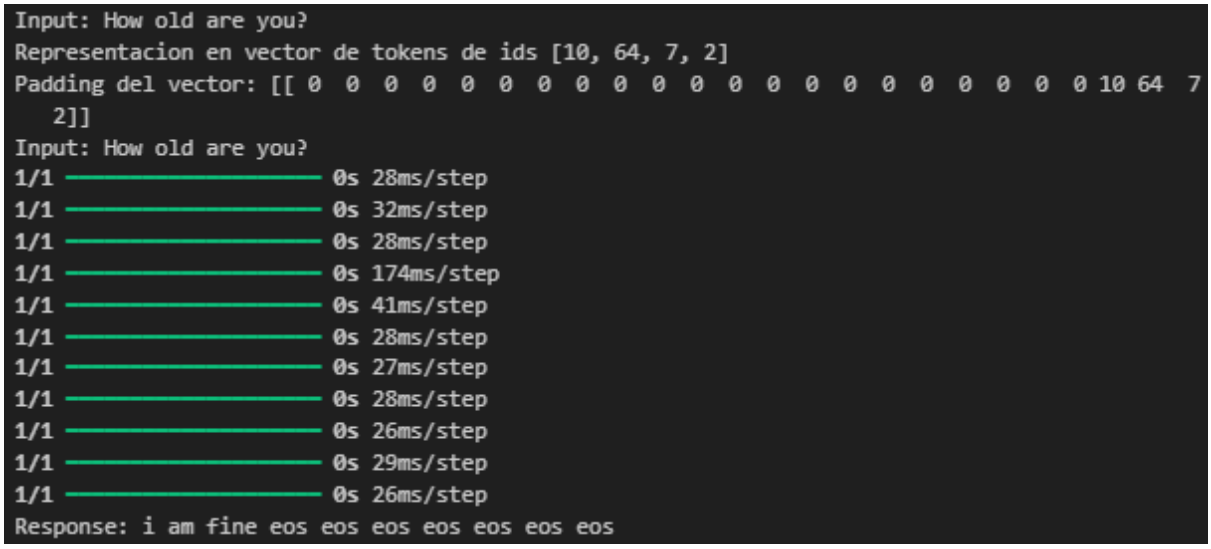
Decoder:



Resultados:

Palabra aleatoria: Input: hi Response: hello how are you eos eos eos eos eos eos

Frase 1: Input: How old are you? Response: i am fine eos eos eos eos eos eos eos



Frase 2: Input: Tomorrow I'm going on vacation Response: i love to read eos eos eos eos eos eos