Artificial intelligence project title: Telegram chat-bot

NLP

NLP: Natural language processing

پردازش زبان طبیعی (NLP) شاخه‌ای از هوش مصنوعی در علوم رایانه است که بر کمک به رایانه‌ها برای درک نحوه نوشتن و صحبت کردن انسان تمرکز دارد.

https://www.youtube.com/watch?v=CMrHM8a3hqw

```
import tensorflow as tf
import tensorflow.keras as keras
import tensorflow.keras.preprocessing as preprocessing
```

- TensorFlow یک کتابخانه نرم افزاری رایگان و open source برای یادگیری ماشین و هوش مصنوعی است. می توان از آن در طیف وسیعی از کارها استفاده کرد، اما تمرکز ویژه ای بر آموزش و استنتاج شبکه های عصبی عمیق دارد.
- tf (tensorflow), keras, preprocessing این سه تا را از کتابخانه TernsorFlow گرفتیم و برای train، مدل کردن و تحلیل مدل استفاده میکنیم.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- Pandas یک کتابخانه پایتون است که برای کار با مجموعه داده ها (data sets) استفاده می شود. دارای عملکردهایی برای تجزیه و تحلیل، تمیز کردن، کاوش و دستکاری داده ها است. نام "Pandas" هم به "Panel Data" و هم به "Python Data Analysis" اشاره دارد و توسط Wes McKinney در سال 2008 ایجاد شد.
- NumPy یک کتابخانه پایتون است که برای کار با آرایه ها (arrays) استفاده می شود. همچنین دارای توابعی برای کار در حوزه جبر خطی (linear algebra)، تبدیل فوریه (Fourier transform) و ماتریس ها (arrays) است. NumPy در سال 2005 توسط Travis Oliphant ایجاد شد.

- Matplotlib یک کتابخانه پایتون است که برای ایجاد نمودارها و نمودارهای دو بعدی (2D graphs) با استفاده از اسکریپت های پایتون استفاده می شود. این کتابخانه دارای ماژولی به نام pyplot است که با ارائه ویژگی هایی برای کنترل line styles ، font properties ، formatting axes و غیره، کار را برای ترسیم آسان می کند.
- کتابخانه pandas و numpy را برای کار با مجموعه داده ها و آرایه ها، استفاده کردیم و برای رسم نمودار ها، ماژول pyplot را از کتابخانه matplotlib گرفتیم.

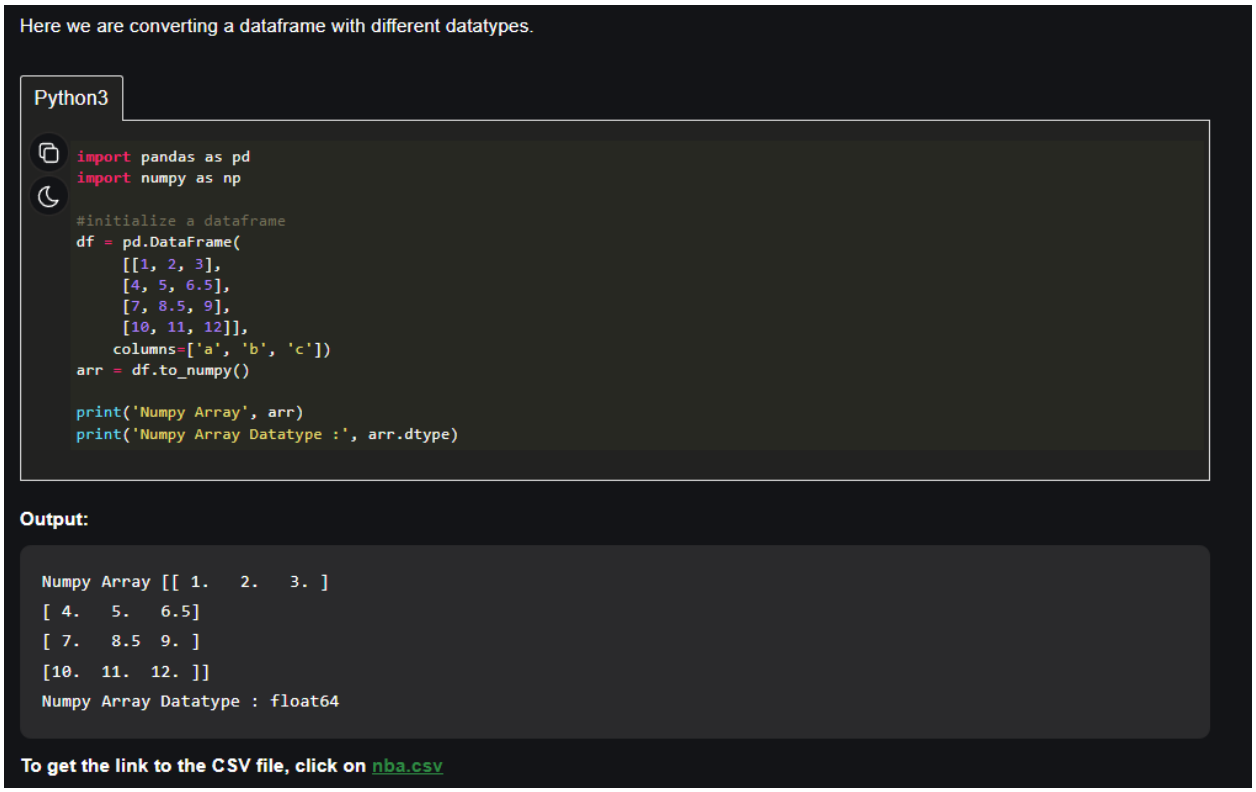
```
data_frame1 = pd.read_csv("/content/1_1_align.csv")
print(data_frame1.shape)
data_frame1.head()
```

- دیتا فریم ها به صورت اکسل های Airline ای است که باید جملات را شناسایی کنیم (tokenization)، و سپس فرآیند word2vec را انجام دهیم یعنی کلمات را به بردار تبدیل کنیم و از بردار ها جهت train کردن استفاده کنیم.
- در خط اول، فایل های csv. را میخوانیم و از کتابخانه pandas برای فریم دادن به آنها استفاده میکنیم و آن ها را در متغیر تعریف شده ی data_frame1 میگذاریم.
- CSV files. : فایل comma-separated values یک فایل متنی محدود شده است که از کاما برای جداسازی مقادیر استفاده می کند. هر خط از فایل یک رکورد داده است. هر رکورد شامل یک یا چند فیلد است که با کاما از هم جدا شده اند.
- خط دوم؛ shape. : این دستور ارگمان های اولیه را نشان میدهد که در اینجا (1952, 12) است که عدد اول سطر و عدد دوم ستون است.
- از ستون های Text و Agreement میخواهیم استفاده کنیم. ستون Agreement نیازی به فرآیند word2vec شدن ندارد زیرا خود متشکل از اعداد صفرو یک است. جملات ستون Text را، که کلمات استفاده شده در این جملات علامت گذاری شده هستند و به صورت تکنیکال از قبل train شده میباشند، متناظر با صفر یا یک بودن Agreement میگذاریم.
- در خط سوم: از دیتا فریم یک head میگیریم و در قسمت بعدی به Train & Test آن میپردازیم.

Start Train & Test data

```
sentences_1 = data_frame1["Text"].to_numpy() # step word2vec
labels_1 = data_frame1["Agreement"].to_numpy()
print(labels_1)
```

- ابتدا باید حالت text را word2vec کنیم.
- DataFrame.to_numpy : DataFrame را به آرایه NumPy تبدیل میکند. به عنوان مثال، اگر نوع داده float16 و float32 باشد، نوع داده نتایج float32 خواهد بود. این ممکن است نیاز به کپی کردن داده ها و مقادیر اجباری داشته باشد که ممکن است هزینه زیادی داشته باشد.



https://www.geeksforgeeks.org/pandas-dataframe-to_numpy-convert-dataframe-to-numpy-array/

- برای خط دوم: همان طور که گفته شد نیازی به فرآیند word2vec برای Agreement نیست زیرا خودش صفر و یک است و فقط باید آن را به صورت numpy دربیاوریم. در نهایت از لیبل ها چاپ میگیریم که داریم: [0. 0. 1. ... nan nan nan] قسمتی صفر و یک داریم و قسمتی هم none (تهی) است که در train لحاظ نمیکنیم.

```
test_split_percent = 0.1

data_length = len(sentences_1)
length_of_trainSet = int((1-test_split_percent)*data_length)
print(data_length)
print(length_of_trainSet)
train_sentences = sentences_1[0 : length_of_trainSet]
test_sentences = sentences_1[length_of_trainSet : ]
# print(test_sentences)
print(len(test_sentences))
train_labels = labels_1[0 : length_of_trainSet]
test_labels = labels_1[length_of_trainSet : ]
```

- حال باید یک دسته بندی انجام دهیم. در قسمت های بعدی به k-fold خواهیم رسید. در اینجا ابتدا 0.1 قسمت اولیه را به عنوان تست در نظر گرفتیم (کل دیتا را به 10 قسمت تقسیم کرده و یکی از ده قسمت را انتخاب کردیم.).
- 1952 سطر داشتیم و در واقع هر سطر یک جمله حساب میشود که همان length جملات ما است.
- 90% تعداد این جملات را به عنوان train انتخاب میکنیم. (test – 1) این عدد همان عددی است که ما برای جداسازی جمله در نظر میگیریم.
- فرآیند split کردن دیتا ها برای test و train را هم برای sentences_1 (که از فرایند word2vec کردن Text ها بدست می آمد) و هم برای label_1 (که از فرآیند word2vec کردن Agreement ها بدست می آمد.) انجام میدهیم.
- از 1952 جمله، 1756 تای آن train و 196 تای آن test میباشد.

Add Parameters

```
# maximum word
vocab_size = 10000
max_sentence_length = 100
# to normalize datasets... number of sentence -> (0, 100)

truncating_0 = 'post'
padding_0 = 'post'
oov_tokens_0 = '@:)'
```

- یک سری Tokenizer ؛ مثل vocab_size یا max_sentence_length که در قسمت Models آوردیم ، برای نرمال سازی به صورت پارامتر در این قسمت میگیریم.

- vocab_size : سقف کل کلمات را 10 هزار در نظر گرفتیم تا از over fit کردن مدل جلوگیری کنیم.
- max_sentence_length : کل جملات را نرمال سازی میکند و در بازه ی صفر تا صد قرار میدهد.
- Truncating : متد truncate() اندازه فایل را به تعداد بایت معین تغییر می دهد. این تغییر از آخر انجام شود.
- Padding : درج کاراکترهای غیر اطلاعاتی در یک رشته به عنوان Padding شناخته می شود. درج را می توان در هر نقطه از رشته انجام داد. اما اینجا ما تعیین کردیم که این درج در آخر انجام شود.
- ’(:@‘ : علامت Tokenizer ما است. به ازای هر word که در کل دیتا موجود نبود این علامت را اختصاص میدهیم.

k-fold Train & Test

```
tokenizer_0 = preprocessing.text.Tokenizer(num_words=vocab_size,
                                   oov_token = oov_tokens_0)
tokenizer_0.fit_on_texts(train_sentences)
# print(tokenizer_0)
word_index = tokenizer_0.word_index
print(word_index)
```

- ما از Tokenizer و padding برای مرتب سازی و تغییر اندازه قاب داده استفاده کرده ایم.
- K-fold انجام شد و حالا باید مجموعه test و train را Tokenize کنیم. یعنی این دو مجموعه را در بالا انتخاب کردیم، حال باید به صورت کلمه ای و برداری در بیاوریم تا بتوانیم در مدل از آن استفاده کنیم.
- از کتابخانه preprocessing استفاده میکنیم تا فرآیند Tokenizer را انجام دهیم. تعداد کلمات را، vocab_size که مقدار 10هزار کلمه بود، قرار میدهیم. همچنین علامت کلماتی که در دیتا موجود نبود را قرار میدهیم.
- fit(). : این دستور را استفاده میکنیم تا تک تک کلمه های درون جملات train را به یک عدد اختصاص دهد. اگر پرینت بگیریم میبینیم که از keras استفاده میکند که خود یک الگوی بزرگ از مجموعه هاست. در آخر index گذاری میکنیم. به این صورت کار میکند که بیشترین کلمه ای که استفاده میشود index 1 دارد که همان مجموعه ی کلمات ناموجود ما است و بعد بیشترین کلمه استفاده شده پس از آن را index 2 میگذارد که در اینجا ’the‘ است و همینطور به صورت نزولی ادامه میدهد.

```
train_sequences = tokenizer_0.texts_to_sequences(train_sentences)
# print(train_sequences)
train_padded = preprocessing.sequence.pad_sequences(
    train_sequences,
    maxlen=max_sentence_length,
```

```
    padding=padding_0,
    truncating=truncating_0
)
```

- : New1