Subject ..... Date .....

طبقه بندی متن با کتابخانه Keras و زبان برنامه نویسی پایتون

۱. مرحله اول آن همیشه تعریف و فراخوانی کتابخانه‌های پایتون است که برای ما پایتون اصلی‌ترین و معروف‌ترین کتابخانه Pandas است.

در هر مرحله نیز می‌توان کتابخانه مورد نیاز را تعریف کرد اما به صورت متداول در ابتدا کتابخانه‌ها تعریف می‌شود.

برای مثال حاضر علاوه بر کتابخانه Pandas سایر کتابخانه‌های زیر نیز مورد نیاز است:

- ✓ SKlearn library
- ✓ Keras library
- ✓ matplot library

۲. مرحله دوم خواندن متون و ذخیره آن در متغیرهای برنامه است که برای ما باشد به شکل نام ستون اطلاعات از فایل متن (text ، csv و tsv و ...) در داخل یک Dataframe ذخیره می‌کند.

```
Python

import pandas as pd

filepath_dict = {'yelp':   'data/sentiment_analysis/yelp_labelled.txt',
                 'amazon': 'data/sentiment_analysis/amazon_cells_labelled.txt',
                 'imdb':   'data/sentiment_analysis/imdb_labelled.txt'}

df_list = []
for source, filepath in filepath_dict.items():
    df = pd.read_csv(filepath, names=['sentence', 'label'], sep='\t')
    df['source'] = source  # Add another column filled with the source name
    df_list.append(df)

df = pd.concat(df_list)
print(df.iloc[0])
```

آدرس ۳ فایل متنی را مشخص می‌کند

یک لیست خالی ایجاد می‌کند

به ازای هر آدرس و source که وجود دارد، فایل مربوطه را می‌خواند و source را هم به آن اضافه می‌کند و در لیست df‌ها ذخیره می‌کند

لیست را تبدیل Dataframe به می‌کند.

برای مثال مطرح شده ۳ داده متنی وجود دارد که هرکدام از قراری جمله و label مرتبط با هر جمله که در این مثال احساس مثبت و منفی است که به ترتیب با ۱ و ۰ نشان داده شده، مشکل شده است.

هر فایل متن خواندن ۳ متن، ایجاد یک Dataframe مشکل از ۳ ستون است که یکی source متن که به ازای Dataframe اضافه می‌شود مشخص می‌شود، متن مربوط به کدام source است.

* نکته مهم این است که این دیتاست پیشنهادی برای Test و Train طبقه‌بندی نشده

Subject ..... Date .....

داریم با یه هردو را در Dataframe قرار دهیم تا بعد از Train و Test تبدیل به آرایه، انجام شود.

که البته لزوماً نیاز به این کار در شبکه‌های عمیق نیست و شبکه pre-train شده را می‌توان به کار برد و در مراحل بعدی از آن استفاده کرد.

۳- مرحله سوم ایجاد مدل یادگیری است. سپس با توجه به نوع شبکه و داده‌ها باید اصلاحات تنظیمی شود.

در برخی موارد ۲۵٪ داده‌ها برای تست در نظر گرفته می‌شود. کتابخانه sklearn این امکان را دارد که با یک خط کد داده‌های تست و یادگیری را جدا کند.

* ایجاد مدل اولیه یا همان baseline Model بسیار در روند بهبود مدل‌های مبتنی بر یادگیری ماشین مهم است.

Python

```
>>> from sklearn.model_selection import train_test_split
>>> df_yelp = df[df['source'] == 'yelp']
>>> sentences = df_yelp['sentence'].values
>>> y = df_yelp['label'].values
>>> sentences_train, sentences_test, y_train, y_test = train_test_split(
...     sentences, y, test_size=0.25, random_state=1000)
```

ماژول split از کتابخانه sklearn عمل تست و یادگیری را انجام می‌دهد

برای مثال با متون ماژل Yelp عملیات انجام شده است

متغیر ستون Sentence داخل object ریخته می‌شود

لیبل‌های مرتبط نیز داخل object با تگ ریخته می‌شود

داده‌های Test و Train و Labelهای مرتبط از هم جدا می‌شوند

۲۵٪ داده‌ست

برای Random Sampling استفاده می‌شود. ولی optional است

۴- مرحله چهارم ایجاد شبکه عصبی با هر جعبه‌بندی است که نیاز است Train شود.

در این مثال هدف یک شبکه عصبی عمیق است.

با توجه به این مثال‌ها ایجاد یک شبکه عصبی (عمیق) است. در این شبکه نامتناظر شبکه‌های عصبی نورون‌های ورودی (لایه ورودی) با بردار ویژگی‌ها (Feature Vector) تغذیه می‌شود و این متناسب به لایه‌های پنهان (hidden layer) می‌رود.

نکته مهم این است که شبکه به وزن (weight) و بایاس (bias) نیاز دارد.

Subject ..... Date .....

در نهایت نیز لایه خروجی است (output layer) که می‌تواند یک یا چند نود داشته باشد.

اگر مسئله دسته‌بندی دودویی باشد ← لایه خروجی یک نود دارد.

اگر مسئله دسته‌بندی کلاسه چند کلاس دارد ← لایه خروجی به تعداد کلاس‌ها نود خواهد داشت.

* تعداد لایه‌های پنهان می‌تواند متغیر و متفاوت باشد که به مسئله بستگی دارد.

- در شبکه عصبی activation function مهم است که انواع مختلفی دارد
  - ReLU ← معمولا برای لایه‌های پنهان
  - Sigmoid ← برای حالت دسته‌بندی
  - SoftMax ← برای حالت چند کلاسی

- همچنین Backpropagation در روند بهینه‌سازی نیز باید برای شبکه تعریف شود.

- در شبکه loss function نیز مهم است که می‌تواند انواع مختلفی داشته باشد
  - cross entropy

- کتابخانه Keras دستورات ساده‌ای برای تعریف شبکه دارد. این کتابخانه بستر Tensorflow و Theano است و دستورات ساده‌تری دارد.

- در مرحله چهارم در واقع شبکه کامل می‌شود تا نیازمندی‌ها استخراج می‌شود و در مرحله پنجم این مدل با کمک کتابخانه ایجاد می‌شود.

- مدل معمولا ساختاری ترتیبی (Sequential) دارد که کتابخانه Keras آن را پشتیبانی می‌کند.

- به طور کلی کتابخانه Keras در مدل اصلی
  - Sequential
  - Functional

را پشتیبانی می‌کند.

در این مثال مدل Sequential است، مدل Sequential لایه‌های شبکه را خواهد داشت.

ابعاد مدل و نوع مدل مهم است و باید برای شبکه تعریف کرد.

* ابعاد ورودی شبکه به ویژگی‌ها و بردار ویژگی دارد. input_dim همان number of features

عبارتی این ابعاد لایه‌های شبکه به ترتیب تعریف می‌شود

Subject ..... Date .....

input_dim => Number of features
model = Sequential()
model.add(Layers_info)

* دستور model.summary() می‌توانه خلاصه اطلاعاتی از شبکه در اختیار ما بگذاره

برای تبدیل داده‌ها به داده‌های قابل فهم توسط ماشین یکی از روش‌های تبدیل آن‌ها به بردار به هر کلمه یک عددی دهیم اینجا که اطلاعات اضافی ندارد و خیلی مناسب نیست

* به دلیل نداشتن اطلاعات اضافی Vectorize کردن زیاد مفید نیست

- تعداد مراحل Train و داده Test باید مشخص شود.

```
Python

>>> from sklearn.feature_extraction.text import CountVectorizer

>>> vectorizer = CountVectorizer()
>>> vectorizer.fit(sentences_train)

>>> X_train = vectorizer.transform(sentences_train)
>>> X_test  = vectorizer.transform(sentences_test)
>>> X_train
<750x1714 sparse matrix of type '<class 'numpy.int64'>'
    with 7368 stored elements in Compressed Sparse Row format>
```

هر جمله را Tokenize می‌کند

به طبقات (دسته‌ها) تبدیل می‌کند. تعداد و طبقه‌ها را در حافظه ذخیره می‌کند

تعداد Sort کردن به هر کلمه یک عدد نسبت می‌دهد.

1- در مرحله بالا جملات داخل object هایی با نام Sentences_train و Sentences_test قرار دادیم

* این کار به نوعی پیش‌پردازش است و با کتابخانه‌هایی مثل nltk نیز قابل انجام است.

حالا داده‌ها آماده ارسال و یادگیری در شبکه است.

```
Python

>>> from keras.models import Sequential
>>> from keras import layers

>>> input_dim = X_train.shape[1]  # Number of features

>>> model = Sequential()
>>> model.add(layers.Dense(10, input_dim=input_dim, activation='relu'))
>>> model.add(layers.Dense(1, activation='sigmoid'))
Using TensorFlow backend.
```

تعداد ورودی را مشخص می‌کند

برداری که با Vectorize ایجاد شده است

ورودی batch size

لایه شبکه مثال ما دو لایه دارد

Subject ..... Date .....

کد دستوری .shape[] ابعاد دوری ما را مشخص می‌کند که برای شبکه ما باید مشخص کنیم.

```
input_dim = X_train.shape[1]
```

* نکته مهم این ابعاد دوری تنها مورد نیاز لایه اول شبکه عصبی است. لایه های دیگر خودکار ایجاد می‌شوند.

- لایه ها یکی پس از دیگری تعریف می‌شوند که در نهایت لایه مشخص است.

تعریف فرایند یادگیری

```
Python
>>> model.compile(loss='binary_crossentropy',
...                optimizer='adam',
...                metrics=['accuracy'])
>>> model.summary()
```

متریک ارزیابی نهایی

خلاصه ساختاری مدل را نشان می دهد

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                17150
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 17,161
Trainable params: 17,161
Non-trainable params: 0
_________________________________________________________________
```

برای شروع فرآیند Traing هم از تعریف مدل و لایه ها، فرآیند یادگیری نیز باید تعریف شود.

حالا دیگر شروع عملیات Training آسان پیش است.

دستور شروع عملیات یادگیری

```
Python
>>> history = model.fit(X_train, y_train,
...                     epochs=100,
...                     verbose=False,
...                     validation_data=(X_test, y_test)
...                     batch_size=10)
```

تعداد Iteration

تعداد Sample هایی که در هر epoch استفاده می‌شود

در نهایت Iterative برای یادگیری داریم.

Subject ..... Date .....

با تکه کد مثلی عملیات Train انجام خواهد شد. حالا به مدل ارزیابی شود.

ارزیابی مدل یا یادگرفته شده

```
Python
>>> loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
>>> print("Training Accuracy: {:.4f}".format(accuracy))
>>> loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
>>> print("Testing Accuracy:  {:.4f}".format(accuracy))
Training Accuracy: 1.0000
Testing Accuracy:  0.7960
```

سپس برای تعریف شبکه، Train و ارزیابی شبکه مراحل زیر انجام شد:

1- Select Model
2- Model add layers (1..N)
3- Compile Model
4- Fit Model
5- Evaluate Model

در مثال فوق متن Text شد و اطلاعات اضافی دیگری نداشت Vectorize

در حالی که Text داده‌های ترتیبی است. عام دوری رحله مثل در مثال مثلی به عنوان یک Vector به شبکه داده شد. (به مدل مثل مثلی Bow (Bag of word) گفته می‌شود.)

روش‌های دیگری هم برای Vectorize کردن داده‌های دوری وجود دارد:

- word as a Vector
- character as a Vector
- N-gram

مثال برای word as a vector مدل one-Hot encode است

که برداری به طول طبقات داریم و هر کلمه یک بردار دارد.

داده‌های غیرعددی به عدد تبدیل می‌شوند تا برای ماشین قابل فهم باشد. one-hot encoding

در این روش ابتدا طبقات را به بردار تبدیل می‌کند و اطلاعات اضافی دیگری از ترتیب کلمات ندارد.

ساده‌ترین روش است.

این روش توسط کتابخانه Sklearn پشتیبانی می‌شود.

بهترین روش برای کاشتن داده‌های غیرعددی که خاصیت ترتیب دارند word embedding است.

Subject ..... Date .....

مثال بعدی با همان داده‌ها انجام با لفت word embedding پیاده سازی می‌شود.

دوره‌برد word embedding داده‌هایی با ابعاد کمتر ایجاد می‌کند اما اطلاعات بیشتری دارند داده‌ها وجود دارد.

بردار طلات به نوعی متراکم می‌شود که به همین خاطر به آن dense word vectors هم گفته می‌شود.

* کتابخانه Keras دستوراتی برای تعریف word embedding دارد اما با توجه به نوع مسئله می‌توانه متفاوت باشد و خود برنامه نویس آن‌ها را تعریف کند.

یادگیری آن کمی یادگیری آسان‌تر است اما ساختار آماری دقیق ایجاد می‌کند که هدف آن شباهت‌ها به یک معنای هندسی است

Map semantic meaning into a Geometric space.

embedding space

دوره‌برد برای پیاده سازی word embedding وجود دارد

در دو دسته عمده ایجاد شود.

- کتابخانه Keras از یکی از دستورات ترتیبی و متن ساده پشتیبانی می‌کند.

به صورت یک مرحله مثل از یادگیری (pre-train) باشد. سپس بصورت جسته درمدل استفاده شود.

دستورات زیر برای یکی از دستورات استفاده می‌شود که دومی متداول‌تر است به‌کارگیری آن در دو دسته عمده است.

اعمال بازیابی طلات و کدگذاری آن‌ها نیز وجود دارد.

```
Python                                                                    >>>
>>> from keras.preprocessing.text import Tokenizer
>>> tokenizer = Tokenizer(num_words=5000)
>>> tokenizer.fit_on_texts(sentences_train)
>>> X_train = tokenizer.texts_to_sequences(sentences_train)
>>> X_test = tokenizer.texts_to_sequences(sentences_test)
>>> vocab_size = len(tokenizer.word_index) + 1  # Adding 1 because of reserved 0 inde
>>> print(sentences_train[2])
>>> print(X_train[2])
Of all the dishes, the salmon was the best, but all were great.
[11, 43, 1, 171, 1, 283, 3, 1, 47, 26, 43, 24, 22]
```

* طلات کدگذاری می‌شود

حداکثر طلات

داده‌ای که Tokenizer قرار است روی آن اعمال شود.

اعمال Tokenizer بر روی داده‌های Train و Test

طول کل طلات موجود

جمله را پیدا می‌کند

تعداد دیکشنری را پیدا می‌کند

Subject ..... Date .....

Python

```
>>> for word in ['the', 'all', 'happy', 'sad']:
...     print('{}: {}'.format(word, tokenizer.word_index[word]))
the: 1
all: 43
happy: 320
sad: 450
```

← بازیابی کد تعیین یافته به طبقات

یک مسئله این است که طبقات داخل هر جمله تعداد متفاوتی دارند و در واقع طول بردار ورودی متفاوت خواهد بود که از pad_sequence استفاده می‌کنیم که ستون های خالی مانده را با صفر پر می‌کند تا طول یکسان شود.

Python

```
>>> from keras.preprocessing.sequence import pad_sequences
>>> maxlen = 100
```

← حداکثر طول

اعمال padding با مقدار صفر روی داده test و Train به منظور هم طول شدن

```
>>> X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
>>> X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)

>>> print(X_train[0, :])
[  1  10   3 282 739  25   8 208  30  64 459 230  13   1 124   5 231   8
  58   5  67   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
```

استفاده از embedding layer در شبکه عصبی هم وجود دارد که با این دستور اجرا می‌شود.

موارد زیر حتماً باید برای آن مشخص شود.

- input_dim → اندازه طبقات
- output_dim → اندازه بردار خروجی
- input_length → اندازه ترتیب (Sequence)

```
model.add(layers.Embedding( ))
model.add(layers.Flattern())
```

word2vec ⟶ Google

Glove ⟶ Stanford NLP Group

Subject ..... Date .....

علاوه بر Vectorize یا ده سازی شده در کتابخانه های قبلی که با word embedding
قابل انجام است که بصورت پیوسته داخل کد Keras نیز وجود دارد
word embedding ها که از قبل محاسبه شده
انجام شده اند استفاده از word embedding های آنان (jointly)
نیز وجود دارد.

word2vec مهم ترین و معروف ترین متد موجود و پیاده سازی شده برای آن وجود دارد و رویکرد
توسعه یافته توسط گوگل است و رویکرد Glove نیز معروف است.

Word2Vector ⟶ Use neural network

Glove ⟶ use co-occurence matrix

* رویکرد word2vec دقیق تر و سریع تر است اما رویکرد Glove راحت تر قابل درک است

* نکته مهم این است که اگر بخواهیم رویکرد خاص دیگری را برای word embedding استفاده
کنیم این امکان توسط کتابخانه Gensim وجود دارد.

برای استفاده از روش های تعریف شده توسط برنامه نویس نیز وجود دارد

پیشنهاد می کنم ابتدا مثالی که در کلاس حل کردیم را بصورت ماتریسی و ورایبل ورودی را هم در نظر بگیریم و در لایه
تعریف لایه های شبکه به عنوان embedding قرار دهیم.

در لایه تعریف ماتریسی با ابعاد Random به عنوان embedding به شبکه داده شده است
و به جای آن برنامه نویس می تواند ماتریس خود را از ورودی و مثل مورد نظر خود بخواند
و در این لایه وارد کند.

Python

```
model = Sequential()
embedding_dim = 50
numpy_matrix = np.random.rand(1747,50)
model.add(layers.Embedding(vocab_size, embedding_dim,
                           weights=[numpy_matrix],
                           input_length=maxlen,
                           trainable=True))
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
history = model.fit(X_train, y_train,
                    epochs=50,
                    verbose=False,
                    validation_data=(X_test, y_test),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

ماتریس Random در لایه embedding به شبکه داده می شود.