به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین چهارم

نام و نام خانوادگی	آرمان فروزش – مهسا ندافی قهنویه
شماره دانشجویی	810100490 - 111946
تاریخ ارسال گزارش	1401.10.05

فهرست

شکل‌ها

جدول‌ها

پاسخ 1. تخمین آلودگی هوا

1-1. سوالات تشریحی

- **Linear interpolation method**

در این روش داده هایی که به اصطلاح گم شده اند و در دسترس نیستند از طریق درون یابی خطی بین دو نقطه(دیتا) به دو روش **forward** یا **backward** جای گذاری میشوند این متد خط بین دو نقطه موجود را در نظر گرفته و نقطه گمشده که میان این دو نقطه وجود دارد را جای گذاری میکند.

$$SL(x) = f(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}} \quad x \in [x_{i-1}, x_i], i = 1, 2, 3, ..., n$$

- **Pearson correlation**

در تحلیل‌های چند متغیره آماری، شیوه‌های مختلف محاسباتی برای اندازه‌گیری وابستگی یا ارتباط بین دو متغیر تصادفی وجود دارد. منظور از ضریب همبستگی بین دو متغیر، قابلیت پیش‌بینی مقدار یکی برحسب دیگری است. یکی از روش‌های نمایش ارتباط بین دو متغیر، محاسبه «کوواریانس» و یا «ضرایب همبستگی» بین آن‌ها است. هر چه مقدار این دو شاخص بزرگتر باشد، نشان‌دهنده ارتباط یا وابستگی بیشتر بین دو متغیر است. یکی از مشهورترین شیوه‌های اندازه‌گیری وابستگی بین دو متغیر کمی، محاسبه ضریب همبستگی پیرسون است. این شاخص توسط «کارل پیرسون» آماردان انگلیسی در سال 1900 معرفی شد.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

باید توجه داشت که ضریب همبستگی پیرسون فقط زمانی که واریانس و امید-ریاضی وجود داشته باشند، قابل محاسبه است.

خصوصیات: بدون واحد بودن ، تقارن ، اندازه گیری ارتباط خطی بین دو متغیر،تعیین جهت همبستگی

- **R_2**

هنگامی که ساختار مدل مشخص شد، از مجموعه آموزشی برای آموزش آن استفاده می شود

شبکه تا همگرایی به منظور ارزیابی کارایی مدل، از شاخص هایی استفاده میکند که **R2** یکی از آنها است.

این ضریب تعیین، نشان دهنده نسبت تمام تغییرات متغیر وابسته است که می تواند توسط متغیر مستقل از طریق رابطه رگرسیون توضیح داده شود. هر چه مقدار **R2** به 1 نزدیکتر شود، متغیر مستقل بهتر می تواند متغیر وابسته را توضیح دهد.

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

1-2. دیتاست

مجموعه داده انتخاب شده در این مقاله (420768 نمونه و 18 ویژگی) از مخزن یادگیری ماشین **UCI** آمده است. این مجموعه داده غلظت آلاینده های هوا و کیفیت هوا را در 12 سایت نشان می دهد. داده های کیفیت هوا از مرکز نظارت بر محیط زیست شهرداری پکن تهیه شده است. داده های هواشناسی که کیفیت هوای هر سایت را نشان می دهد با نزدیکترین ایستگاه هواشناسی منطقه مطابقت دارد.

تمامی این دیتا توسط کتابخانه **Pandas** فراخوانی شده است:

```
csvs = glob.glob('/content/drive/MyDrive/Colab Notebooks/PRSA_Data_20130301-20170228/**.csv')
print("There are totatl {} csv files in the given dataset".format(len(csvs)))
```

```
There are totatl 12 csv files in the given dataset
```

1-3. پیش پردازش

بدین منظور برای یک دست شده دیتا ابتدا تمامی ستون های سایت **Aotuzhongxin** و **PM2.5** در بقیه 11 سایت را بصورت دیتافریم در کنار یکدیگر قرار دادیم. و ستون های اضافه زیر نیز حذف شدند:

```
['No', 'year', 'month', 'day', 'hour', 'SO2', 'NO2', 'O3', 'station']
```

	PM2.5	PM10	CO	TEMP	PRES	DEWP	RAIN	wd	WSPM	PM2.5-Changping	PM2.5-Dingling	PM2.5-Dongsi	PM2.5-Guanyuan	PM2.5-Gucheng	PM2.5-Huairou	PM2.5-Nongzhanguan	PM2.5-Shunyi	PM2.5-Tiantan	PM2.5-Wanliu	PM2.5-Wanshouxigong
0	4.0	4.0	300.0	-0.7	1023.0	-18.8	0.0	337.5	4.4	3.0	4.0	9.0	4.0	6.0	7.0	5.0	3.0	6.0	8.0	9.0
1	8.0	8.0	300.0	-1.1	1023.2	-18.2	0.0	0.0	4.7	3.0	7.0	4.0	4.0	6.0	4.0	8.0	12.0	6.0	9.0	11.0
2	7.0	7.0	300.0	-1.1	1023.5	-18.2	0.0	337.5	5.6	3.0	5.0	7.0	3.0	5.0	4.0	3.0	14.0	6.0	3.0	8.0
3	6.0	6.0	300.0	-1.4	1024.5	-19.4	0.0	315.0	3.1	3.0	6.0	3.0	3.0	6.0	3.0	5.0	12.0	6.0	11.0	8.0
4	3.0	3.0	300.0	-2.0	1025.2	-19.5	0.0	0.0	2.0	3.0	5.0	3.0	3.0	5.0	3.0	5.0	12.0	5.0	3.0	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35059	12.0	29.0	400.0	12.5	1013.5	-16.2	0.0	315.0	2.4	28.0	11.0	16.0	13.0	14.0	16.0	14.0	27.0	20.0	11.0	11.0
35060	13.0	37.0	500.0	11.6	1013.6	-15.1	0.0	292.5	0.9	12.0	13.0	18.0	20.0	27.0	21.0	18.0	47.0	11.0	15.0	13.0
35061	16.0	37.0	700.0	10.8	1014.2	-13.3	0.0	315.0	1.1	7.0	9.0	23.0	16.0	22.0	17.0	15.0	18.0	18.0	13.0	14.0
35062	21.0	44.0	700.0	10.5	1014.4	-12.9	0.0	337.5	1.2	11.0	10.0	23.0	11.0	9.0	11.0	11.0	18.0	15.0	12.0	12.0
35063	19.0	31.0	600.0	8.6	1014.1	-15.9	0.0	22.5	1.3	20.0	13.0	30.0	15.0	12.0	11.0	10.0	15.0	15.0	7.0	13.0

35064 rows × 20 columns

شکل 1-1 دیتاست پس از انجام مراحل 1-3-1و 1-3-2

1-3-1. Missing Value

در حین مرتب سازی داده ها مقادیر گمشده را با استفاده از **linear interpolation** با استفاده از فرمول زیر جای گذاری کردیم:

```
df = df.interpolate(method ='linear', limit_direction ='forward') # I
nsertion of missing values by Linear Interpolation method
```

با استفاده از کد زیر تعداد داده های Nan هر ستون را نمایش میدهیم:

```
data.isna().sum()
```

```
PM2.5                  0
PM10                   0
CO                     0
TEMP                   0
PRES                   0
DEWP                   0
RAIN                   0
wd                     0
WSPM                   0
PM2.5-Changping        0
PM2.5-Dingling         0
PM2.5-Dongsi           0
PM2.5-Guanyuan         0
PM2.5-Gucheng          0
PM2.5-Huairou          0
```

```
PM2.5-Nongzhanguan      0
PM2.5-Shunyi            0
PM2.5-Tiantan           0
PM2.5-Wanliu            0
PM2.5-Wanshouxigong     0
dtype: int64
```

2-3-1. Encoding Categorical Variable

در این بخش همانطور که در مقاله ذکر شده بود 16 جهت باد را با فواصل 22.5 درجه به درجه تبدیل کردیم.

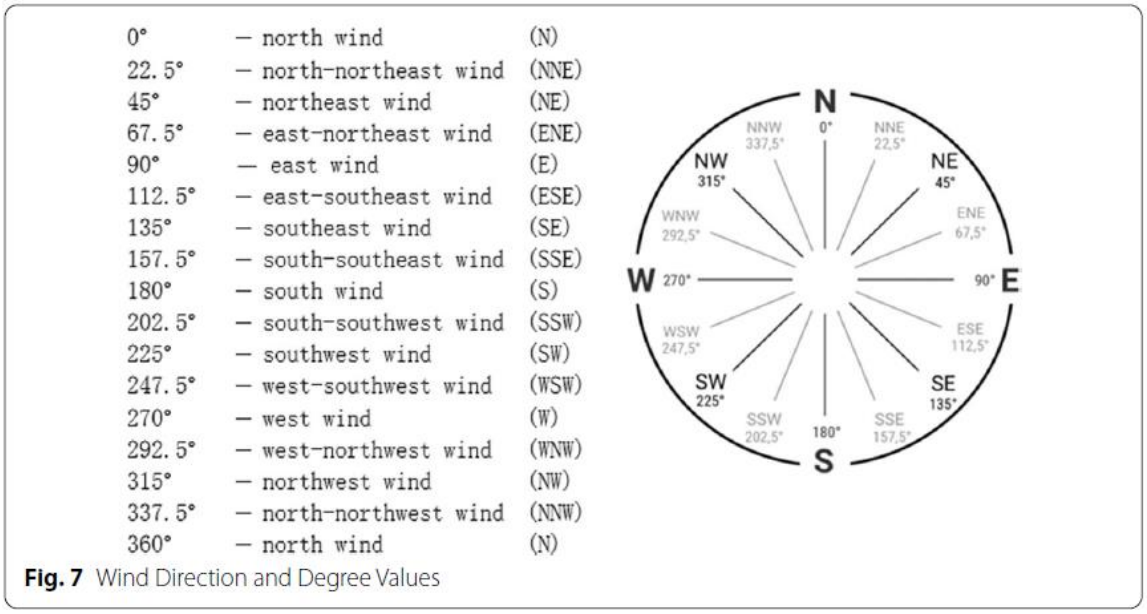


Fig. 7 Wind Direction and Degree Values

شکل 1-2 تبدیل جهت به درجه

```
['N', 'NNE', 'NE', 'ENE', 'E', 'ESE', 'SE', 'SSE', 'S', 'SSW', 'SW',
'WSW', 'W', 'WNW', 'NW', 'NNW']
```

```
 [0.0, 22.5, 45.0, 67.5, 90.0, 112.5, 135.0, 157.5, 180.0,
202.5, 225.0, 247.5, 270.0, 292.5, 315.0, 337.5]
```

کد تمامی قسمت های بالا در زیر دیده میشود و دیتای حاصل از این کد در شکل 1-1 قابل مشاهده است:

```
dataFrames = []
for i in range(12):
```

```
    df = pd.read_csv(csvs[i], index_col=None, header=0) #read each CS
V file
    df = df.interpolate(method ='linear', limit_direction ='forward')
 # Insertion of missing values by Linear Interpolation method
#-------------------------
collect PM2.5 from all dataset and important column from Aotizhongxin
------------------------------
    if i == 0 :
      #print(df['wd'].unique())
#-----------------------------------------
Encoding Categorical Variable-------------------------------------
-------------
      WD = ['N', 'NNE', 'NE', 'ENE', 'E', 'ESE', 'SE', 'SSE'
            , 'S', 'SSW', 'SW', 'WSW', 'W', 'WNW', 'NW', 'NNW']
      degree = []
      for i in range(len(WD)):
        degree.append (i*22.5)
      print(WD,'\n',degree)
      for i in range(len(WD)):
        df['wd'] = df['wd'].replace(to_replace=WD[i] ,value=degree[i]
)# replace wd with degree values
#-----------------------------------------Missing value----------
-----------------------------------------------------
      df = df.interpolate(method ='linear', limit_direction ='forward
') # Insertion of missing values by Linear Interpolation method
      dataFrames.append(df)
    else :
      df = df.interpolate(method ='linear', limit_direction ='forward
') # Insertion of missing values by Linear Interpolation method
      newname = 'PM2.5-' + df.iloc[1,17]
      df = df.rename(columns={ 'PM2.5': str(newname) })
      dataFrames.append(df[newname])
data = pd.concat(dataFrames,axis=1)
data = data.drop(columns=['No', 'year', 'month', 'day', 'hour', 'SO2'
, 'NO2', 'O3', 'station'])#delete unnecessary columns
```

3-3-1 Nomarlization

نرمالیزیشن **MinMaxScaler** برای تمامی ستون های دیتاست اجرا شده است با استفاده از کد زیر اجرا شده است:

```
    for feature in data.columns:
  maxf = data[feature].max()
  minf = data[feature].min()
  data[feature] = (data[feature] - minf) / (maxf - minf) #minmax Norm
alization
```

باید دقت شود که داده های train و test باید جداگانه نرمال شوند اما با توجه به اینکه در مقاله اشاره ای به این موضوع نشده بود از آن گذر میکنیم.

4-3-1. Pearson Correlation

هنگامی که ویژگی های زیادی برای ورود به شبکه برای آموزش وجود دارد، یافتن همبستگی بین مقدار خروجی هدف و آن ویژگی ها باعث کاهش پیچیدگی آموزش و بهبود عملکرد می شود.

```
#---------------------------------------- Pearson Correlation -------
--------------------------------------------------------------------
--------
corr=datac.corr(method = 'pearson').round(2)
heatmap=corr.drop(['PM10', 'CO', 'TEMP', 'PRES', 'DEWP', 'RAIN', 'wd'
, 'WSPM']).drop(['PM10', 'CO', 'TEMP', 'PRES', 'DEWP', 'RAIN', 'wd',
'WSPM'],axis=1)
heatmap.style.background_gradient(cmap='OrRd')
```

نتایج در شکل زیر قابل مشاهده است:

	PM2.5	PM2.5-Changping	PM2.5-Dingling	PM2.5-Dongsi	PM2.5-Guanyuan	PM2.5-Gucheng	PM2.5-Huairou	PM2.5-Nongzhanguan	PM2.5-Shunyi	PM2.5-Tiantan	PM2.5-Wanliu	PM2.5-Wanshouxigong
PM2.5	1.000000	0.840000	0.820000	0.950000	0.950000	0.900000	0.850000	0.940000	0.890000	0.930000	0.940000	0.920000
PM2.5-Changping	0.840000	1.000000	0.910000	0.820000	0.840000	0.850000	0.850000	0.810000	0.810000	0.810000	0.870000	0.790000
PM2.5-Dingling	0.820000	0.910000	1.000000	0.800000	0.820000	0.830000	0.850000	0.790000	0.800000	0.780000	0.840000	0.770000
PM2.5-Dongsi	0.950000	0.820000	0.800000	1.000000	0.960000	0.900000	0.830000	0.960000	0.880000	0.960000	0.930000	0.950000
PM2.5-Guanyuan	0.950000	0.840000	0.820000	0.960000	1.000000	0.920000	0.850000	0.950000	0.880000	0.950000	0.950000	0.950000
PM2.5-Gucheng	0.900000	0.850000	0.830000	0.900000	0.920000	1.000000	0.860000	0.880000	0.860000	0.890000	0.930000	0.890000
PM2.5-Huairou	0.850000	0.850000	0.850000	0.830000	0.850000	0.860000	1.000000	0.820000	0.890000	0.820000	0.850000	0.810000
PM2.5-Nongzhanguan	0.940000	0.810000	0.790000	0.960000	0.950000	0.880000	0.820000	1.000000	0.880000	0.950000	0.920000	0.940000
PM2.5-Shunyi	0.890000	0.810000	0.800000	0.880000	0.880000	0.860000	0.890000	0.880000	1.000000	0.880000	0.870000	0.870000
PM2.5-Tiantan	0.930000	0.810000	0.780000	0.960000	0.950000	0.890000	0.820000	0.950000	0.880000	1.000000	0.920000	0.960000
PM2.5-Wanliu	0.940000	0.870000	0.840000	0.930000	0.950000	0.930000	0.850000	0.920000	0.870000	0.920000	1.000000	0.920000
PM2.5-Wanshouxigong	0.920000	0.790000	0.770000	0.950000	0.950000	0.890000	0.810000	0.940000	0.870000	0.960000	0.920000	1.000000

شکل 3-1 ضرایب همبستگی پیرسون

5-3-1. Feature selection

در نهایت بهترین ویژگی ها که ستون ها **PM2.5** برای تمامی دیتاست ها و **PM10,CO,TEMP,PRES,DEWP,RAIN,wd,WSPM** مربوط به **Aotizhongx** در **data** نگهداره شده و در فایل **20Features.csv** ذخیره میشود.

```
#-------------------------------------
- Save 20 important features as 20features.csv file ----------------
-----------------------------------------
filepath = Path('/content/drive/MyDrive/Colab Notebooks/20features.cs
v')
filepath.parent.mkdir(parents=True, exist_ok=True)
data.to_csv(filepath)
```

همچنین 20 درصد از دیتا به عنوان داده های تست و 80 درصد به عنوان داده های ترین انتخاب شده است. که توسط **train_test_split** با **shuffle=False** قابل دستیابی است.

1-3-6. Supervised dataset

در این بخش دیتای موجود بدین صورت **supervised** میشود که برای **lag** یک روز دیتای صفرم شامل ساعت 0 تا 23 و **PM2.5** ساعت 24 لیبل این دیتا را تشکیل میدهد بدین ترتیب دیتای اول شامل ساعات 1 تا 24 و لیبل **PM2.5** ساعت 25 است.

برای **lag** 7 روز ،دیتای صفرم شامل ساعات 0 تا 167 و لیبل **PM2.5** ساعت 168 است و ....

کد :

- لگ یک روز

```
atac = data.values
d=np.zeros((35040,24,20)) # split to 1day lag
label = []

for i in range(35040):
    label.append(datac[i+24,0])
    d[i,0:24,:] = datac[i:i+24,:]

label = np.array(label)
X_train, X_test, y_train, y_test = train_test_split(d, label, test_si
ze=0.2, shuffle=False)
print(X_train.shape)
print(y_train.shape)
(28032, 24, 20)
(28032,)
```

- لگ 7 روز

```
datac = data.values
d=np.zeros((34896,168,20)) # split to 1day lag
label = []

for i in range(34896):
    label.append(datac[i+168,0])
    d[i,0:168,:] = datac[i:i+168,:]

label = np.array(label)
X_train, X_test, y_train, y_test = train_test_split(d, label, test_si
ze=0.2, shuffle=False)
```

```
#X_train = Scaler.fit_transform(X_train[:,:,all])
#X_test = Scaler.transform(X_test)
#test_size = 7008

#datac_train = d[:-test_size]
#datac_test = d[-test_size:]
#label_train = label[:-test_size]
#label_test = label[-test_size:]


print(X_train.shape)
print(y_train.shape)
(27916, 168, 20)
(27916,)
```

4-1. آموزش شبکه

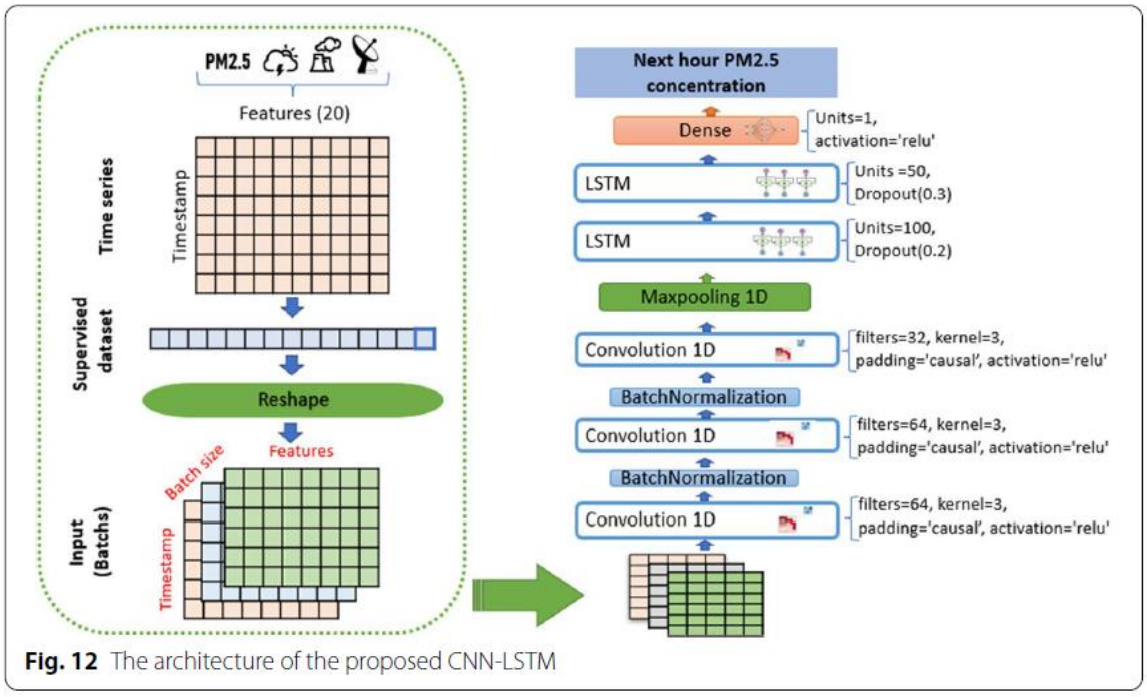


Fig. 12 The architecture of the proposed CNN-LSTM

شکل 4-1 شبکه مورد استفاده در مقاله

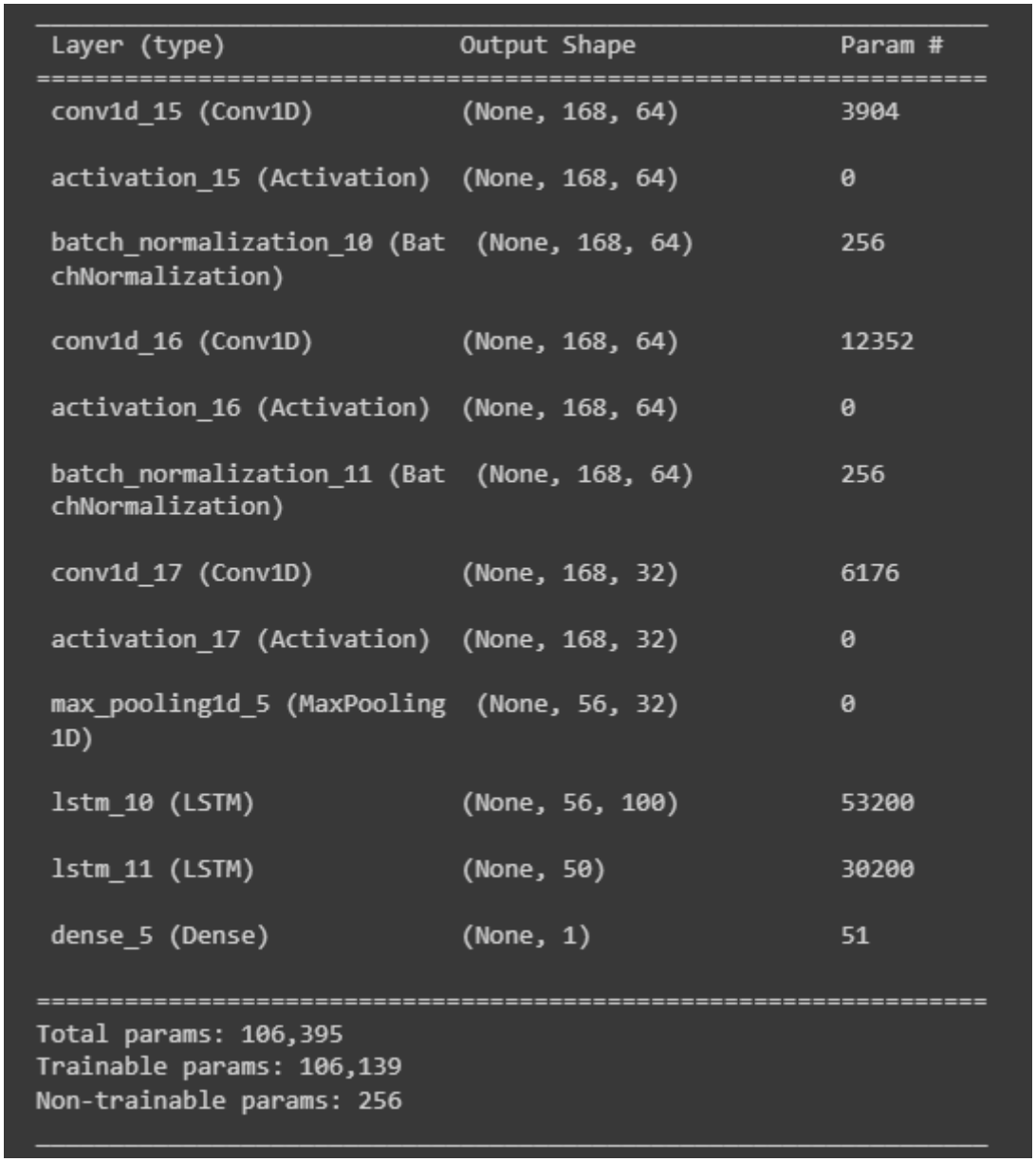
شبکه ی مورد استفاده در مقاله شبکه ی **CNN-LSTM** می باشد که ابتدا 3 لایه کانولوشن و سپس دولایه **LSTM** در خود دارد :

```
model = Sequential()

# 1st Conv-Block
model.add(Conv1D(64, kernel_size=3, input_shape = (168,20), padding='
causal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
```

```
# 2nd Conv-Block
model.add(Conv1D(64,kernel_size=3, padding='causal'))
model.add(Activation('relu'))
model.add(BatchNormalization())

# 3rd Conv-Block
model.add(Conv1D(32,kernel_size=3, padding='causal'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3))

#LSTM blockes
model.add(LSTM(100, dropout=0.2 , return_sequences=True))
model.add(LSTM(50, dropout=0.3))

# Classification Block
model.add(Dense(1 , activation='relu'))


model.summary()
```

ورودی این شبکه برای **lag1 => (24*20)** و **lag1 => (168*20)** می باشد.

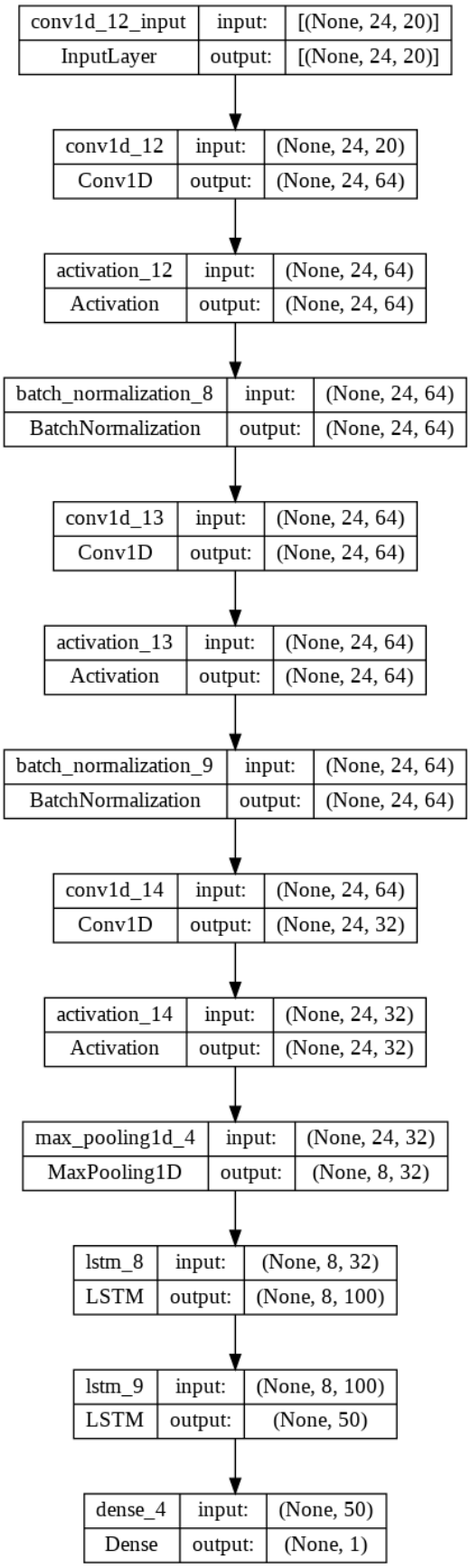
بهترین پارامترها **loss = mse , batchsize=32 , epoch=200** طبق گفته مقاله انتخاب شده است.



Layer (type)	Output Shape	Param #
conv1d_15 (Conv1D)	(None, 168, 64)	3904
activation_15 (Activation)	(None, 168, 64)	0
batch_normalization_10 (BatchNormalization)	(None, 168, 64)	256
conv1d_16 (Conv1D)	(None, 168, 64)	12352
activation_16 (Activation)	(None, 168, 64)	0
batch_normalization_11 (BatchNormalization)	(None, 168, 64)	256
conv1d_17 (Conv1D)	(None, 168, 32)	6176
activation_17 (Activation)	(None, 168, 32)	0
max_pooling1d_5 (MaxPooling1D)	(None, 56, 32)	0
lstm_10 (LSTM)	(None, 56, 100)	53200
lstm_11 (LSTM)	(None, 50)	30200
dense_5 (Dense)	(None, 1)	51

Total params: 106,395
Trainable params: 106,139
Non-trainable params: 256

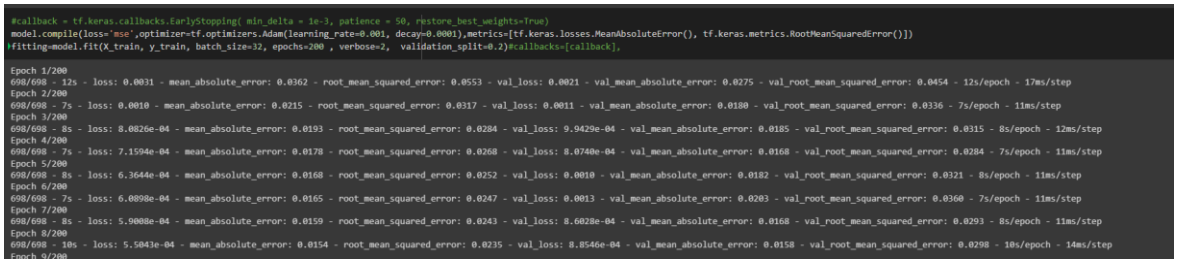
شکل 1-5 خلاصه ای از **Model**



شکل 1-6 لایه های شبکه برای **lag1** در کد

در نهایت برای گزارش مقادیر خواسته شده از کد زیر استفاده شد:

```
callback = tf.keras.callbacks.EarlyStopping( min_delta = 1e-
3, patience = 50, restore_best_weights=True)
model.compile(loss='mse',optimizer=tf.optimizers.Adam(learning_rate=0
.001, decay=0.0001),metrics=[tf.keras.losses.MeanAbsoluteError(), tf.
keras.metrics.RootMeanSquaredError()])
fitting=model.fit(X_train, y_train, batch_size=32, epochs=200 , verbo
se=2,  validation_split=0.2, callbacks=[callback])

predicted_y = model.predict(X_test).flatten()
print("R2: " + str(RSquare()(y_test, predicted_y).numpy()))
print("MAE: " + str(tf.keras.losses.MeanAbsoluteError()(y_test, predi
cted_y).numpy()))
print("RMSE: " + str(tf.keras.metrics.RootMeanSquaredError()(y_test,
predicted_y).numpy()))
```



شکل 1-7 شبکه در حال آموزش برای lag7

- نتایج **lag1**

```
R2: 0.9354158
MAE: 0.014311018
RMSE: 0.023742381
```

- نتایج **lag7**

```
219/219 [==============================] - 2s 5ms/step
R2: 0.93712753
MAE: 0.014506426
RMSE: 0.023458716
```

خروجی شبکه **PM2.5**برای ساعت بعدی می باشد که تست شده و توسط معیار **R2**گزارش میشود، باید دقت شود که مقادیر **MAE,RMSE** به دلیل اینکه داده ها نرمالایز شده است کوچک شده اند و اگر دیتا را بعد از خروج از شبکه به مقادیر اصلی خود بارگردانیم و با استفاده از آنها مقادیر راحساب کنیم به مقادیر اصلی دست پیدا میکنیم. به همین جهت **callback** را بی اثر میکنیم تا به دقت بهتری دست پیدا کنیم و شبکه در ایپاک 53 یا ایپاک دیگری متوقف نشود.

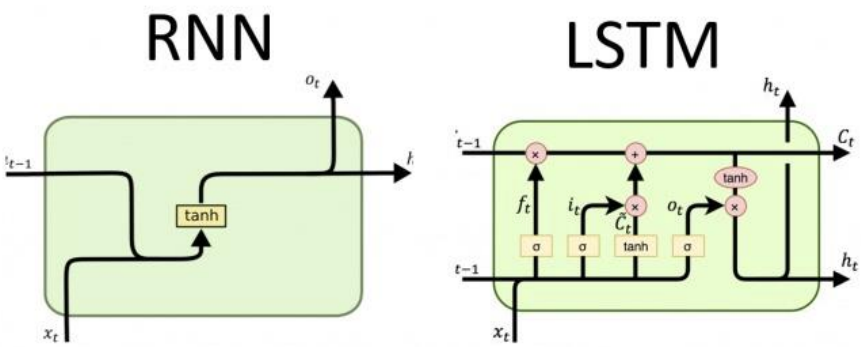
در اینجا معیار **R2** به درستی نزدیک به 1 بدست آمده است که نشان میدهد مدل به خوبی آموزش دیده شده است و با در نظر گرفته **epoch** بیشتر میتوان به دقت بیشتری در این مورد دست یافت.

در واقع تعداد ایپاک به این دلیل کم در نظر گرفته شده است چرا که مقادیر **loss** در مقاله به علت سر و کار داشتن با مقادیر اصلی خروجی نه مقادیر نرمالایز شده بزرگ بوده است.

پاسخ 2. تشخیص اخبار جعلی

2-۱. توضیحات مدل‌ها

شبکه های عصبی مکرر (RNN) از جمله شبکه های مطرح در زمینه پردازش داده های متوالی (sequential) هستند که از ویژگی های مهم آنها داشتن حافظه و به خاطر سپردن ورودی است که که امر مهمی در پردازش داده های متوالی و سریزمانی است. شبکه های LSTM یکی از انواع شبکه های RNN هستند که از مهمترین نفاوت های LSTM میتوان به ساختار پیچیده تر آن اشاره کرد. همچنین به دلیل وجود گیت های موجود در معماریش، جریان اطلاعاتی بهتری وجود دارد و بر خلاف مدل های اولیه RNN که مشکل vanishing gradients داشتند، این مشکل را تا حد بسیار زیادی حل کرده باشند.



شکل 1.2 ساختار بلوک **LSTM** و **RNN**

با توجه به اینکه این مدل از شبکه ها در پردازش داده های متوالی و سری زمانی عملکرد بسیار خوبی دارند. به علت اینکه در داده‌های متنی از لحاظ معنایی و گرامری وابستگی و توالی بین کلمات وجود دارد، استفاده از شبکه ای که بتواند بصورت متوالی این پردازش را انجام دهد باسیار با ارزش است، در نتیجه استفاده از شبکه های RNN و LSTM در پردازش داده های متنی بسیار مطلوب خواهد بود.

روش ارائه شده در مقاله مربوطه، استفاده از شبکه hybrid که ترکیبی از شبکه LSTM و CNN است، میباشد. از آنجایی که شبکه های CNN توانایی بالای در استخراج ویژگی های داده های ورودی دارند، تفاوت اصلی که این مدل ارائه شده با شبکه های بازگشتی عادی دارد همین استخراج ویژگی ها قبل از ورود اطلاعات به شبکه LSTM است که با توجه به اینکه این مسئله یک مسئله دسته بندی داده ها بین داده های جعلی و واقعی است، باعث افزایش کارایی مدل میشود. بصورت کلی در این مدل شبکه

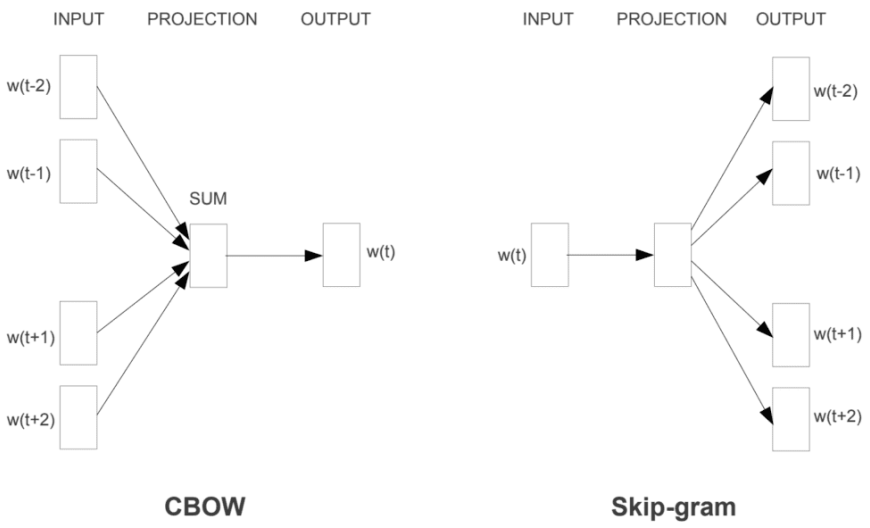
CNN ویژگی‌های محلی (local features) را استخراج کرده و در اختیار LSTM برای آموزش وابستگی های متنی بین داده ها قرار میدهد.

2-2. ورودی مدل

Word embedding یک نوع نمایش برای داده های متنی است که در آن کلمات یکه معنی یکسانی دارند، نمایش یکسانی نیز دارند. Word embedding در واقع تکنیکی است که در آن هر کلمه از متن به صورت جداگانه به صورت یک بردار در فضای از پیش تعریف شده، نگاشت می‌شود و این بردارهای عددی قابلیت استفاده در شبکه های عصبی را دارند چراکه شبکه های عصبی نیاز به ورودی های عددی برای پردازش دارند. سه تا از مطرح ترین روش های word embedding عبارتند از: Embedding Layer ، Word2Vec و GloVe .

Embedding Layer) در این روش یک شبکه عصبی برای انجام word embedding آموزش داده میشود. سپس از این لایه در ابتدای شبکه مورد نظر استفاده می‌شود.

Word2Vec) این روش یک روش آماری برای embedding کارآمد یک کلمه، مستقل از پیکره متنی است. این روش در سال 2013 توسط google ساخته شد. دو مدل مختلف یادگیری که میتوان از آنها در رویکرد Word2Vec استفاده کرد نیز معرفی شدند که میتوان از آنها برای embedding استفاده کرد که در شکل 2.1 مشاهده میشوند.



شکل 2.2 مدل های آموزش مدل **Word2Vec**

GloVe) الگوریتم Global Vectors for Word Representation یا GloVe، در واقع یک extention بر روی روش word2vec برای یادگیری موثر بردارهای کلمه است که توسط پنینگتون در دانشگاه استنفورد توسعه داده شده است. این روش رویکردی برای تطبیق آمار جهانی تکنیک‌های فاکتورسازی ماتریس مانند LSA با یادگیری مبتنی بر زمینه محلی در word2vec است. GloVe به جای استفاده از یک پنجره برای تعریف بافت محلی، یک ماتریس صریح متن کلمه یا هم‌روی کلمه با استفاده از آمار در کل مجموعه متن ایجاد می‌کند. نتیجه یک مدل یادگیری است که ممکن است منجر به جاسازی کلمات به طور کلی بهتر شود.

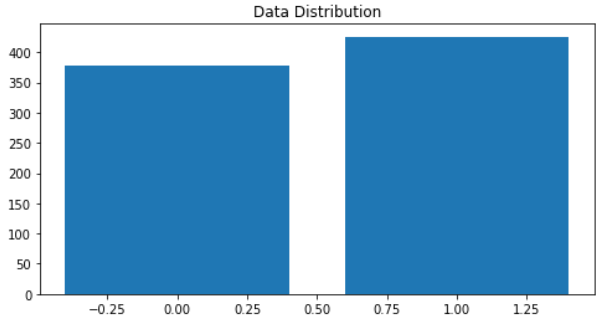
در این مسئله از روش GloVe برای Word Embedding ورودی ها استفاده کرده‌ایم.

2-3. پیاده سازی

2-3-1. پیش پردازش

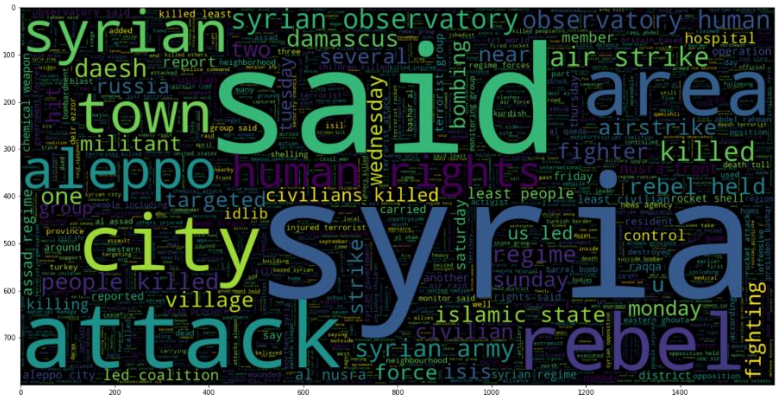
با توجه به روش ارائه شده در مقاله ابتدا پس از دریافت دیتاست، علائم و متن های اضافه همچون URL را از متون مربوط به ستون های article_content و article_title را با استفاده از کتابخانه re پاک میکنیم. در نتیجه متن های بدست آمده تنها شامل کلمات است. سپس با استفاده از SnowballStemmer از کتابخانه NLTK کلمات موجود را stem میکنیم، با این کار، کلمات را به ریشه‌ی مصدریشان تغییر میدهیم. این کار باعث میشود که کلمات هم ریشه بصورت یکسانی شوند و در ادامه پس از embedding بردار مربوطه آنها یکسان شود. سپس در ادامه یک ستون جدید به دیتاست اضافه میکنیم که شامل داده های دو ستون article_content و article_title هستند. با این کار میتوانیم از این ستون ایجاد شده به عنوان ورودی و از ستون labels به عنوان خروجی استفاده کنیم.

برای بررسی بیشتر دیتاست، ابتدا نمئدار کلاس های مربوطه موجود را رسم میکنیم. کلاس 0 مربوط به اخبار جعلی و کلاس 1 مربوط به اخبار حقیقی است.

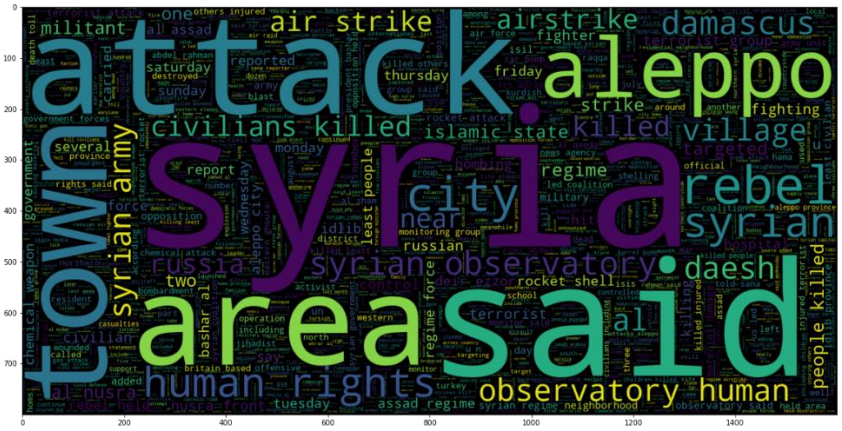


شکل 3.2 توزیع داده ها در دو کلاس جعلی(0) و حقیقی(1)

با توجه به شکل 3.2 ، مشاهده میشود که دیتاست بالانس است. سپس برای درک بهتر از داد ها با استفاده از کتابخانه WordCloud کلامت پرتکرار موجود در هر کلاس را به نمایش میگذاریم.



شکل 4.2 کلمات موجود در اخبار حقیقی



شکل 5.2 کلمات موجود در اخبار جعلی

سپس داده های موجود در دیتاست را با نسبت 80 به 20 به دو دسته train و test تقسیم میکنیم. دیتاست train پس از انجام پیش پردازش در شکل 6.2 مشاهده میشود. در نهایت ما میخواهیم با دادن داده های ستون texts به شبکه(x)، به عنوان خروجی labels را (y) دریافت کنیم.

	unit_id	article_title	article_content	source	date	location	labels	texts
99	1923102909	air raid kills 19 syrians held village	date publication 5 october 2016 air raid hit v...	alaraby	10/5/2016	aleppo	1	air raid kills 19 syrians held village date pu...
24	1918150028	six killed aleppo bombing evacuation 4 syrian ...	posted april 19 2017 asharq al awsat english s...	asharqalawsat	4/19/2017	aleppo	1	six killed aleppo bombing evacuation 4 syrian ...
307	1924058066	air strikes kill 4 syria truce zone monitor jo...	last updated nov 022017 beirut syrian governme...	jordantimes	11/2/2017	damascus	1	air strikes kill 4 syria truce zone monitor jo...
442	1926479005	deadly russian airstrikes kill 35 destroy hosp...	tuesday 31 may 2016 18 14 deadly russian airst...	etilaf	5/31/2016	idlib	1	deadly russian airstrikes kill 35 destroy hosp...
525	1926479240	syrian army islamic state clash near army airp...	july 18 2014 syrian army islamic state clash n...	reuters	7/18/2014	homs	0	syrian army islamic state clash near army airp...

شکل 6.2 دیتاست پس از انجام پیش پردازش

سپس با استفاده از Tokenizer از Keras و فیت کردن آن روی داده های train، به هر یک از کلامت موجود در جمله های ورودی، یک مقدار عددی متناسب با تعداد دفعات تکرارشان در جمله اختصاص داده میشود. با توجه به اینکه مقاله حداکثر طول این بخش را 100 در نظر گرفته است، اعداد اختصاص داده شده به کلمات بین 1تا100 هستند و از عدد 0 نیز برای padding کردن استفاده میشود، به اینصورت که اگر در جمله‌ای تعداد کلمات کمتر از 100 بود، بقیه بردار را 0 بگذارد، به اینصورت ابعاد داده ها نیز همه یک بردار 100 تایی خواهند بود.

```
[ ]  from keras.preprocessing.text import Tokenizer

     tokenizer = Tokenizer()
     tokenizer.fit_on_texts(train_data.texts)

     word_index = tokenizer.word_index
     vocab_size = len(tokenizer.word_index) + 1
     print("Vocabulary Size :", vocab_size)

     Vocabulary Size : 9216
```

شکل 7.2 کد **tokenization**

همانگونه که در شکل 7.2 مشاهده میشود در کل داده هاس train، تعداد 9216 کلمه وجود دارد که tokenize شده اند. پس از اعمال tokenization برروی هردو دیتاست train و test ابعاد x_train و x_test بصورت زیر خواهد شد.

```
y_train shape: (643, 1)        Training X Shape: (643, 100)
y_test shape: (161, 1)         Testing X Shape: (161, 100)
```

شکل 8.2 ابعاد داده های شبکه

حال این داده ها آماده ورود به شبکه هستند.

2-3-2. آموزش مدل‌ها

در این مسئله دو مدل RNN و hybrid(CNN-RNN) را مورد بررسی قرار میدهیم. در هر دو مدل لایه اول را به Embedding اختصاص میدهیم. ما در این مسئله از embedding از پیش آموزش داده شده GloVe استفاده میکنیم.

برای ایجاد این لایه از دستور Embedding از Keras استفاده کرده و بعد embedding را نیز 300(با توجه یه مقاله) قرار میدهیم و برای این شبکه از وزن های آماده GloVe استفاده میکنیم.

```
embedding_layer = tf.keras.layers.Embedding(vocab_size,
                                            300,
                                            weights=[embedding_matrix],
                                            input_length=100,
                                            trainable=False)
```

شکل 9.2 لایه embedding

با توجه به گفته مقاله، هر یک از مدل‌ها را 10 ایپاک و با batch size=64 آموزش میدهیم و از ارزیابی accuracy برای آنها استفاده میکنیم.

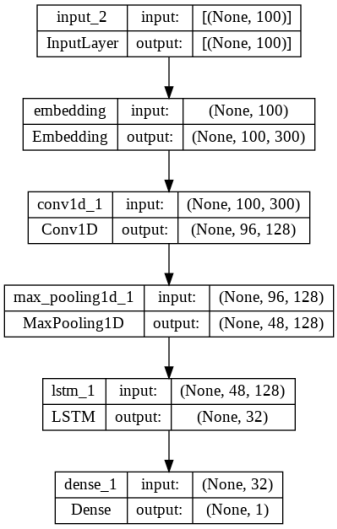
در آموزش هردو شبکه از بهینه ساز Adam با نرخ یادگیری 0.000001 و با توجه به اینکه این شبکه یک binary classification میباشد از تابع هزینه binary cross entropy استفاده می‌شود.

1) مدل Hybrid

برای ساخت این مدل از مدل ارائه شده در مقاله استفاده میکنیم. ساختار شبکه در شکل 10.2 مشاهده می‌شود.

```
sequence_input = Input(shape=(100,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = Conv1D(128, 5, activation='relu')(embedding_sequences)
x = MaxPooling1D(2)(x)
x = LSTM(32, activation=None)(x)
outputs = Dense(1, activation='sigmoid')(x)
model = Model(sequence_input, outputs)
model.summary()
```

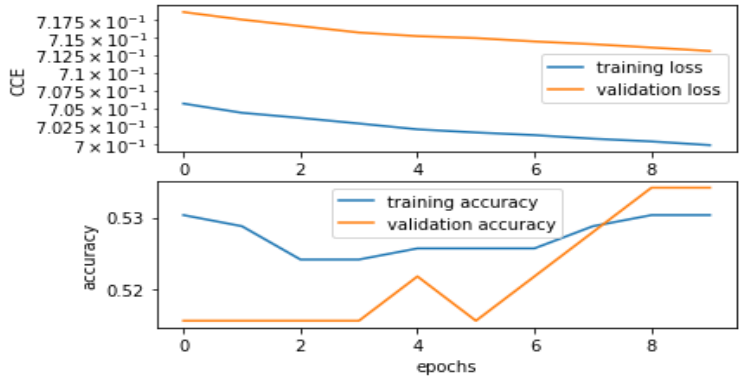
```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 100)]             0

 embedding (Embedding)       (None, 100, 300)          2764800

 conv1d_2 (Conv1D)           (None, 96, 128)           192128

 max_pooling1d_2 (MaxPooling  (None, 48, 128)          0
 1D)

 lstm_2 (LSTM)               (None, 32)                20608

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 2,977,569
Trainable params: 212,769
Non-trainable params: 2,764,800
```



شکل 10.2 ساختار شبکه **Hybrid**

نتایج بخش آموزش شبکه Hybrid را در شکل11.2 مشاهده می‌کنید.

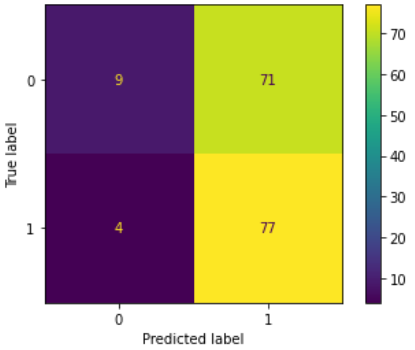
```
Epoch 1/10
11/11 [==============================] - 4s 242ms/step - loss: 0.7056 - accuracy: 0.5303 - val_loss: 0.7186 - val_accuracy: 0.5155
Epoch 2/10
11/11 [==============================] - 2s 152ms/step - loss: 0.7043 - accuracy: 0.5288 - val_loss: 0.7175 - val_accuracy: 0.5155
Epoch 3/10
11/11 [==============================] - 2s 153ms/step - loss: 0.7036 - accuracy: 0.5241 - val_loss: 0.7166 - val_accuracy: 0.5155
Epoch 4/10
11/11 [==============================] - 2s 159ms/step - loss: 0.7028 - accuracy: 0.5241 - val_loss: 0.7157 - val_accuracy: 0.5155
Epoch 5/10
11/11 [==============================] - 2s 156ms/step - loss: 0.7020 - accuracy: 0.5257 - val_loss: 0.7151 - val_accuracy: 0.5217
Epoch 6/10
11/11 [==============================] - 2s 155ms/step - loss: 0.7016 - accuracy: 0.5257 - val_loss: 0.7149 - val_accuracy: 0.5155
Epoch 7/10
11/11 [==============================] - 2s 155ms/step - loss: 0.7012 - accuracy: 0.5257 - val_loss: 0.7144 - val_accuracy: 0.5217
Epoch 8/10
11/11 [==============================] - 2s 221ms/step - loss: 0.7007 - accuracy: 0.5288 - val_loss: 0.7140 - val_accuracy: 0.5280
Epoch 9/10
11/11 [==============================] - 3s 309ms/step - loss: 0.7003 - accuracy: 0.5303 - val_loss: 0.7135 - val_accuracy: 0.5342
Epoch 10/10
11/11 [==============================] - 3s 306ms/step - loss: 0.6998 - accuracy: 0.5303 - val_loss: 0.7130 - val_accuracy: 0.5342
```



شکل 11.2 نتیجه آموزش شبکه شبکه **Hybrid**

همانگونه که در شکل 11.2 مشاهده می‌شود Validation loss روند نزولی مناسبی داشته و از طرفی نیز validation accuracy نیز روند صعودی داشته که نشان دهنده این است سیستم به خوبی آموزش دیده است.

پس از اتمام آموزش، بهترین مدل از لحاظ val_loss را ذخیره کرده و از آن برای ارزیابی داده های test استفاده می‌کنیم.



	precision	recall	f1-score	support
0	0.69	0.11	0.19	80
1	0.52	0.95	0.67	81
accuracy			0.53	161
macro avg	0.61	0.53	0.43	161
weighted avg	0.61	0.53	0.43	161

شکل 12.2 نتیجه شبکه **Hybrid** روی داده های **test**

در نهایت مشاهده می‌شود که این شبکه به دقت 53 درصد دست یافته است.

2) مدل RNN

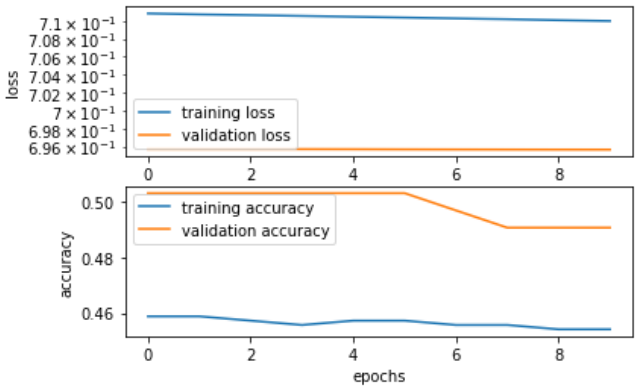
برای ساخت این مدل از یک لایه LSTM با سایز 32 که هم سایز با LSTM به کار رفته در شبکه hybrid می‌باشد استفاده می‌کنیم. ساختار این شبکه در شکل 13.2 قابل مشاهده می‌باشد.

```
Model: "model_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_13 (InputLayer)       [(None, 100)]             0

 embedding (Embedding)       (None, 100, 300)          2704200

 lstm_12 (LSTM)              (None, 32)                42624

 dense_32 (Dense)            (None, 1)                 33

=================================================================
Total params: 2,746,857
Trainable params: 42,657
Non-trainable params: 2,704,200
_________________________________________________________________
```

شکل 13.2 ساختار شبکه **RNN**

نتایج بخش آموزش شبکه RNN را در شکل14.2 مشاهده میکنید.

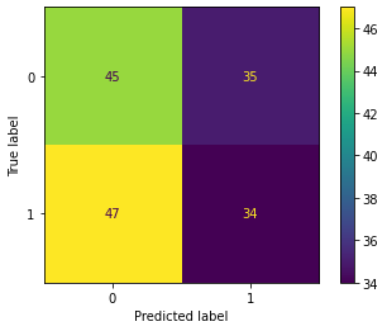
```
Epoch 1/10
11/11 [==============================] - 3s 133ms/step - loss: 0.7108 - accuracy: 0.4588 - val_loss: 0.6957 - val_accuracy: 0.5031
Epoch 2/10
11/11 [==============================] - 1s 102ms/step - loss: 0.7107 - accuracy: 0.4588 - val_loss: 0.6957 - val_accuracy: 0.5031
Epoch 3/10
11/11 [==============================] - 1s 102ms/step - loss: 0.7107 - accuracy: 0.4572 - val_loss: 0.6957 - val_accuracy: 0.5031
Epoch 4/10
11/11 [==============================] - 1s 99ms/step - loss: 0.7106 - accuracy: 0.4557 - val_loss: 0.6957 - val_accuracy: 0.5031
Epoch 5/10
11/11 [==============================] - 1s 104ms/step - loss: 0.7105 - accuracy: 0.4572 - val_loss: 0.6957 - val_accuracy: 0.5031
Epoch 6/10
11/11 [==============================] - 1s 102ms/step - loss: 0.7104 - accuracy: 0.4572 - val_loss: 0.6957 - val_accuracy: 0.5031
Epoch 7/10
11/11 [==============================] - 1s 101ms/step - loss: 0.7103 - accuracy: 0.4557 - val_loss: 0.6957 - val_accuracy: 0.4969
Epoch 8/10
11/11 [==============================] - 1s 100ms/step - loss: 0.7102 - accuracy: 0.4557 - val_loss: 0.6957 - val_accuracy: 0.4907
Epoch 9/10
11/11 [==============================] - 1s 100ms/step - loss: 0.7101 - accuracy: 0.4541 - val_loss: 0.6957 - val_accuracy: 0.4907
Epoch 10/10
11/11 [==============================] - 1s 98ms/step - loss: 0.7100 - accuracy: 0.4541 - val_loss: 0.6957 - val_accuracy: 0.4907
```



شکل 14.2 نتیجه آموزش شبکه **RNN**

همانگونه که مشاهده می‌شود این loss شبکه نسبت به شبکه hybrid با سرعت بسیار کمتری روند نزولی داشته و به خوبی train نشده. یکی از دلایل مهم این اتفاق، بزرگ بودن ابعاد ورودی به لایه LSTM است، چرا که در حالت hybrid ورودی شبکه ابتدا وارد لایه CNN شده و فیچرمپ هایی با ابعاد کوچکتر ایجاد شده که باعث کاهش حجم محاسباتی در لایه LSTM شده و باعث یادگیری بهتر شبکه می‌شود.

در نهایت نتایج ارزیابی این شبکه برروی داده های test را در شکل 15.2 مشاهده می‌کنید.



	precision	recall	f1-score	support
0	0.49	0.56	0.52	80
1	0.49	0.42	0.45	81
accuracy			0.49	161
macro avg	0.49	0.49	0.49	161
weighted avg	0.49	0.49	0.49	161

شکل 15.2 نتیجه شبکه **RNN** روی داده های **test**

در نهایت می‌بینیم که دقت این شبکه به 49 درصد رسیده است. برای مقایسه بهتر نتایج نهایی دو شبکه را در جدول 1.2 مشاهده می‌کنید.

جدول 1.2 مقایسه نتایج **Hybrid** و **RNN**

	Accuracy	Precision	Recall	F1 score
Hybrid	53	61	53	43
RNN	49	49	49	49

با توجه به بالانس بودن داده‌ها، معیار accuracy می‌تواند معیار مناسبی برای ارزیابی شبکه‌ها باشد، همانگونه که مشاهده می‌شود، دقت شبکه hybrid 4درصد از شبکه RNN بیشتر است.

4-2. تحلیل نتایج

یکی از دلایل مهمی که باعث کمتر بودن دقت شبکه RNN نسبت به شبکه Hybrid بوده است، بزرگ بودن ابعاد ورودی به لایه LSTM است، چرا که در حالت hybrid ورودی شبکه ابتدا وارد لایه CNN شده و فیچرمپ هایی با ابعاد کوچکتر ایجاد شده که باعث کاهش حجم محاسباتی در لایه LSTM شده و باعث یادگیری بهتر شبکه می‌شود.

برای بهبود در شبکه ها نیز میتوان از regularization های بییشتر در مدل ها همچون dropout و یا BatchNorm استفاده کرد. از طرفی نیز با توجه به حجم کم داده ها در این دیتاست، با کاهش BatchSize میتوان شبکه را بهتر آموزش داد. استفاده از Bidirectional LSTM به جای LSTM نیز میتواند باعث بهبود عملکرد شبکه شود اما این کار حجم محاسباتی را بالا برده و زمان مورد نیاز برای train شبکه را افزایش می‌دهد.