به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

نام و نام خانوادگی	پرهام بیچرانلو – آناهیتا هاشم زاده
شماره دانشجویی	۸۱۰۱۰۰۳۰۳ - ۸۱۰۱۰۰۵۰۲
تاریخ ارسال گزارش	۱۴۰۱/۱۰/۹

درس شبکه‌های عصبی و یادگیری عمیق

تمرین چهارم

فهرست

شکل‌ها

جدول‌ها

پاسخ ۱. عنوان پرسش اول به فارسی

۱-۱. سوالات تشریحی

- **Linear interpolation method:** یک روش برای پر کردن مقادیر تهی یا null جدول است. در این روش مقدار گمشده از اتصال نقاط قبلی و بعدی با یک خط راست بدست می‌آید. به طور دقیق‌تر از فرمول زیر استفاده می‌کند:

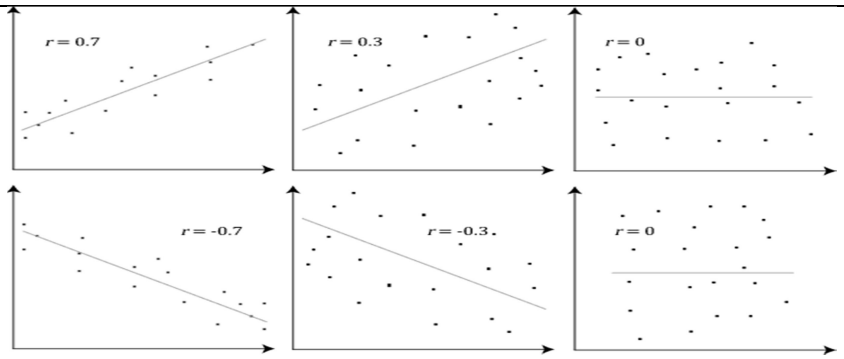
$$y(x) = y1 + (x - x1)\frac{(y2 - y1)}{(x2 - x1)}$$

این روش بیشتر هنگام کار با داده‌های سری زمانی استفاده می شود زیرا در داده‌های سری زمانی ما دوست داریم مقادیر گم شده را با یک یا دو مقدار قبلی پر کنیم. برای مثال، فرض کنید دما، اکنون ما همیشه ترجیح می دهیم دمای امروز را با میانگین ۲ روز گذشته پر کنیم، نه با میانگین ماه. همچنین می توانیم از درون یابی برای محاسبه میانگین متحرک استفاده کنیم.

- **Pearson correlation:** فرض کنید X و Y دو متغیر تصادفی هستند که دارای امید ریاضی E(X) و E(Y) و واریانس V(X) و V(Y) هستند. ضریب همبستگی بین X و Yرا با P(X,Y) نشان داده و به صورت زیر محاسبه می‌شود:

$$\mathrm{P(X,Y)} = corr(X,Y) = \frac{E[(x - E(X))(Y - E(Y))]}{[V(X)V(Y)]^{\frac{1}{2}}}$$

هر چه قدر مقدار ضریب همبستگی به ۱ یا ۱- نزدیک شود، وجود رابطه خطی بین دو متغیر بیشتر می‌شود. اگر مثبت باشد رابطه دو متغیر مستقیم است و اگر منفی باشد، رابطه آن‌ها معکوس خواهد شد. اگر هم نزدیک صفر باشد یعنی آن دو متغیر مستقل از هم هستند. چند مثال در شکل زیر آمده است:



شکل ۱. انواع میزان همبستگی دو متغیر

- **R^2**: ضریب تعیین (R-squared correlation) میزان ارتباط خطی بین دو متغیر را اندازه گیری می کند. R^2 نسبت تغییرات متغیر وابسته را که می توان به متغیر مستقل نسبت داد اندازه گیری می کند. در تعاریف موجود بهR^2 ، ضریب تعیین یا ضریب تشخیص نیز گفته می شود. به بیان ساده می توان گفت ضریب تعیین نشان می دهد که چند درصد تغییرات متغیر های وابسته در یک مدل رگرسیونی با متغیر مستقل تبیین می شود. فرمول آن به صورت زیر است:

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})}{\sum(y_i - \bar{y})^2}$$

هر چقدر به یک نزدیک باشد یعنی اگر از متغیرهای مستقل استفاده کنید هیچ خطایی وجود ندارد که این بهترین حالت است. اگر به صفر نزدیک باشد یعنی متغیرهای مستقل هیچ تاثیری روی برآورد خط رگرسیون ندارند.

۱-۲. دیتاست

از محیط Kaggle برای استفاده از GPU استفاده کردیم. برای همین داده رو هم از خود کگل برداشتیم که با نام Beijing Multi-Site Air-Quality Data Set در این سایت قرار دارد. که یک فایل زیپ است. و شامل ۱۲ فایل csv است که هرکدام مروبط به یک ایستگاه است. ابتدا از کتابخانه glob برای بدست آوردن آدرس هر کدام استفاده کریم سپس از یک حلقه استفاده کردیم و هر کدام را با کتابخانه pandas و دستور read_csv خواندیم. و آن‌ها را به هم چسباندیم در نهایت همه‌ی اطلاعات در دیتافریم data ذخیره شد.

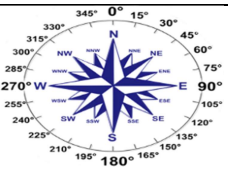
۱-۳. پیش پردازش

۱-۳-۱. Missing value

با دستور data.isnull().sum() تعداد null های هر ستون را می‌شماریم که می‌بینیم برای خیلی از ستون‌ها تعداد زیادی خونه خالی داریم. که این برای تحلیل ما مشکل ساز می‌شود برای همین باید آن‌ها را با روشی که مقاله گفته یعنی Linear interpolation پر کنیم. که کافیست از دستور data.interpolate(method = ‘linear’) استفاده کنیم.

۱-۳-۲. Encoding Categorical Variable

برای این قسمت طبق شکلی که در مقاله آمده است عمل می‌کنیم.



شکل ۲. تبدیل جهت جغرافیایی به درجه

که برای این کار از یک دیکشنری که کلیدها جهت جغرافیایی هستند و مقادیر درجه آن کلید هستند استفاده می‌کنیم. و از دستور data.replace({‘WinDir’: dict}) که dict همان دیکشنری ما است استفاده می‌کنیم.

۱-۳-۳. Nomarlization

برای این کار از فرمول زیر استفاده می‌کنیم:

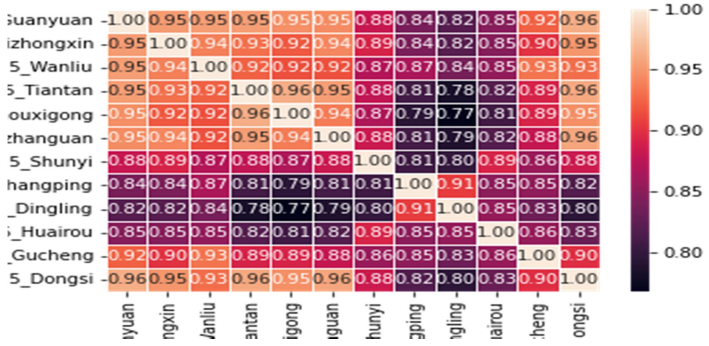
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

یعنی یک حلقه روی جدول می‌زنیم و هربار برای هر ستون مقدار بیشینه و کمینه را بدست می‌آوریم و بعد با توجه به آن داده‌های آن ستون را نرمال می‌کنیم. که مقادیر بین صفر تا یک قرار می‌گیرند. این کمک می‌کند بازه فیچرها یکسان شود تا پیچیدگی داده‌ها کمتر شود.

فقط نکته انیجاست که مقدار ماکسیم و مینیمم برای ستون ’PM2.5‘ را ذخیره می‌کنیم تا بعد از آموزش مدل بتوانیم پیش بینی را به مقدار اصلی بازگردانیم.

۱-۳-۴. Pearson Correlation

یک heatmap است. برای این کار ابتدا یک دیتافریم جدید می‌سازیم که هر ستون آن شامل مقادیر ’PM2.5‘ یکی از سایت‌ها باشد. سپس از دستور sns.heatmap که از کتابخونه seaborn فراخوانی شده استفاده می‌کنیم که ورودی آن کرولویشن دیتافریم جدید است. یعنی new_df.corr(). در واقع هیت مپ ما کورولیشن سایت‌های مختلف با هم را نشان می‌دهد که در شکل زیر خروجی آن آورده شده است:



شکل ۳. کورولیشن بین **PM2.5** سایت‌ها

همانطور که در شکل بالا می‌بینید این کورلویشن(همبستگی) بین سایت‌ها برای PM2.5 زیاد است. پس یعنی می‌توانیم از PM2.5 های دیگر سایت‌ها PM2.5 سایت Aotixhongxin را تخمین زد.

۱-۳-۵. Feature selection

برای این کار ابتدا جدولی که شامل ستون‌های کربن دی اکسید، دما، بارندگی، جهت باد و... است از جدول data می‌سازیم. که به این نحو است که ابتدا سطرهایی که ستون سایت آن‌ها Aotizhongxin است را انتخاب کرده و سپس ستون‌های گفته شده را انتخاب کرده. و اسم جدول را df3 می‌گذرایم

حال جدولی که در مرحله قبل درست کردیم که شامل PM2.5 سایت‌های مختلف در هر ستون بود را با جدول جدیدی که شامل ستون‌های کربن دی اکسید، دما، بارندگی، جهت باد و... است با هم ترکیب می‌کنیم. از دستور pd.concat([new_df, df3], axis = 1) استفاده می‌کنیم. نتیجه آن با دستور df.to_csv(‘new_data.csv’) در اکسل با نام ’new_data‘ ذخیره می‌شود که در فایل‌ها ضمیمه شده است.

۱-۳-۶. Supervised dataset

در اینجا باید توجه کنیم که داده‌های سری زمانی داده آموزش و داده‌های تست کنار هم باشند تا مدل درست کار کند. برای همین اگر ابتدا شافل کنیم بعد بین این دو تقسیم کنیم اشتباه است چون مثلا ممکن است در داده آموزش از یک روز مشخص مثلا ۱۰ ساعتش را که کنار هم نیستند را داشته باشیم که نمی‌توان با آن‌ها درست پیش بینی کرد چون کلی گپ داریم.

برای همین به سادگی داده را از یک نقطه به دو قسمت تقسیم می‌کنیم که ۸۰ درصد اول برای داده ترین باشد و ۲۰ درصد دیگر برای داه تست. برای این کار از دستور df_train = dfs[: split_point] و df_test = dfs[split_point:] استفاده می‌کنیم.

اما این تمام ماجرا نیست. حال باید داده‌های آموزش و تست را برای خودشون هم جداسازی کنیم. به اینطورت که برای مدل ۲۴ ساعته، ۲۴ ساعت را به عنوان داده ورودی و ۱ ساعت بعدی را به عنوان تارگت در نظر بگیریم. روند زیر را در نظر بگیرید:

Input: [d0-h0,d0–h1, d0-h2,….,d0-h23] → target: [d1-h0]

Input: [d0-h1,d0–h2, d0-h3,….,d1-h0] → target: [d1-h1]

Input: [d0-h2,d0–h3, d0-h4,….,d1-h1] → target: [d1-h2]

…

که d به معنای روز است و h به معنای ساعت. اینطوری به تعداد ساعت‌هایی که داده جمع کردیم یک زوج مرتب ورودی و تارگت داریم. که شیپ داده‌های train و test برای لگ یک روزه به صورت زیر می‌باشد:

X_train.shape: (28478, 24, 20)

y_train.shape: (28478, 1)

X_test.shape: (6538, 24, 20)

y_test.shape: (6538, 1)

برای پیش بینی با لگ هفت روزه باید داده‌های ۷ روز که هرکدام شامل ۲۴ ساعت است یعنی ۱۶۸ ساعت ر بدهیم و ساعت ۱۶۹ام را پیش بینی کنیم. و یکی جلو برویم. که ابعاد داده‌های آموزش و تست آن به شکل زیر می‌شود:

X_train.shape: (28334, 168, 20)

y_train.shape: (28334, 1)

X_test.shape: (6394, 168, 20)

y_test.shape: (6394, 1)

که برای داده train بعد اول تعداد داده‌ها، بعد دوم تعداد ساعت‌ها، بعد سوم تعداد فیچرها را نشان می‌دهد.

برای داده test بعد اول تعداد داده‌ها، و بعد دوم هم ساعتی که می‌خواهیم پیش بینی کنیم است.

این کار را با استفاده از تابع split_sequence انجام می‌دهیم.

بعد از این چون از پایتورچ استفاده کردیم باید مدل را به دیتالودرها بدهیم. ابتدا داده‌ها را تبدیل به تسنور کرده و سپس از TensorDataset(train_data,test_data) برای جفت کردن فیچرها و تارگت استفاده می‌کنیم و خروجی آن را به DataLoader می‌دهیم.

۱-۴. آموزش شبکه

شبکه را با ساختاری که مقاله داده است می‌سازیم:

```
Cnn_Lstm(
  (conv1d1): Conv1d(20, 64, kernel_size=(3,), stride=(1,), padding=(2,))
  (conv1d2): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(2,))
  (conv1d3): Conv1d(64, 32, kernel_size=(3,), stride=(1,), padding=(2,))
  (batchnorm1d): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (maxpooling1d): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  (lstm1): LSTM(32, 100, batch_first=True)
  (dropout1): Dropout(p=0.2, inplace=False)
  (lstm2): LSTM(100, 50, batch_first=True)
  (dropout2): Dropout(p=0.3, inplace=False)
  (relu): ReLU(inplace=True)
  (linear): Linear(in_features=50, out_features=1, bias=True)
)
```

که این پیاده سازی با پایتورچ انجام شده است. مدل در کلاسی به نام Cnn_Lstm تعریف شده است.

هدف از لایه‌های کانولوشن: برای استخراج ویژگی از فیچرهای خام استفاده می‌شود. که روی فیچرها یعنی مقادیر PM2.5 و سایر فیچرها حرکت می‌کند. اگر روی سری زمانی حرکت می‌کرد معنادار نبود!

هدف از لایه maxPooling: انتخاب فیچرهای مهمتر و جلوگیری از بیش برازش.

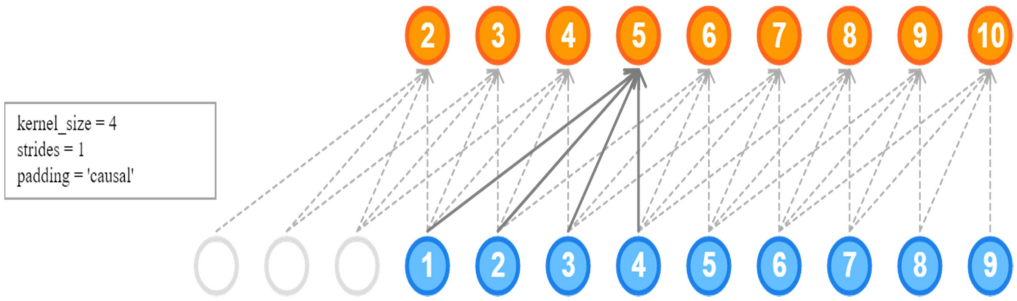
هدف از لایه Lstm: یادگرفتن میزان وابستگی‌های زمانی در سری زمانی.

هدف از لایه فولی کانکتت: استفاده از خروجی lstm برای بدست آوردن مقدار PM2,5.

نکته مهم: در لایه‌های کانولوشنی از پدینگ casual استفاده کرده. که عموما در سری زمانی به کار می‌رود. به اینصورت که به ابتدا دیتا صفر اضافه می‌کند. تا بتوان مقادیر را در همان تایم‌های اولیه هم پیش بینی کرد. چون پایتورچ این پدینگ رو نداره از فرمول آن استفاده کردم و دستی پدینگ رو ۲ قرار

دادم. البته چون از هر دو طرف پد می‌دهد در آخر با slicing دو خانه آخر را دور می‌ریزیم. چون هدف ما پد دادن به خانه‌هایی ابتدایی بوده نه خانه‌های انتهایی!

شکل زیر عملکرد این نوع پدینگ را نشان می‌دهد:

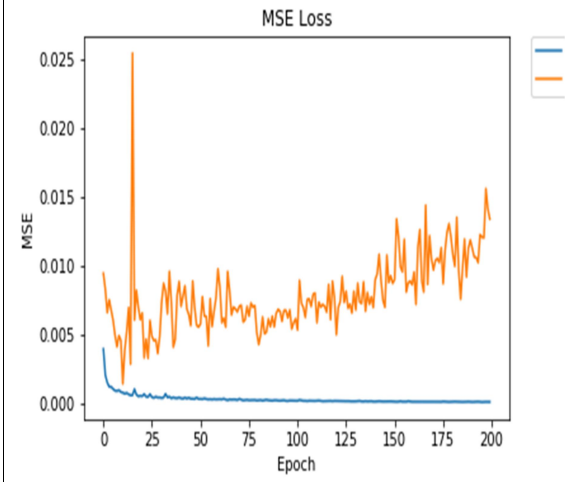


شکل ۴. پدینگ causal

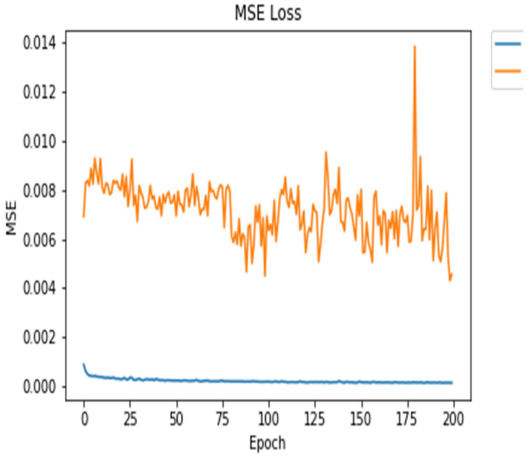
از بهینه ساز آدام با هایپر پارامترهای گفته شده در مقاله استفاده کردیم. و طبق گفته مقاله از تابع هزینه MSE برای آموزش استفاده شده است.

ورودی مدل همان دیتالودرهایی که در مرحله قبل گفتیم است. توجه کنید که دوبار باید مدل را آموزش دهیم یکبار برای لگ یک روزه و یکبار برای لگ هفت روزه.

مدل را برای هر کدام ۲۰۰ ایپاک اجرا کردیم که مقدار loss در شکل زیر آوردیم.



شکل ۵. نمودار mse loss برای لگ یک روزه

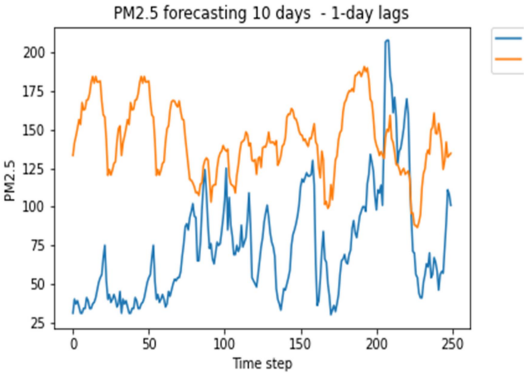


شکل ۶. نمودار mse loss رای لگ هفت روزه

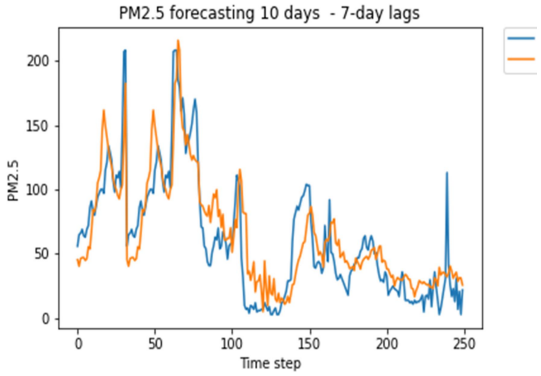
همانطور که مشاهده می‌شود مقدار loss برای داده آموزشی کاهش می‌یابد اما برای دیتای valid اینطوری نیست و نوسانی است. چند علت می‌توان برای آن متصور شد. اول اینکه مقدار loss در بازه

10^{-3} تغییر می‌کند که مقدار ناچیزی است. دوم اینکه شاید به این دلیل باشد که در مقاله فقط مقادیر PM2.5 نرمال شده است اما من با توجه به صورت سوال بقیه فیچرها رو هم نرمال کردم. البته بعید است مشکل این باشد چون نرمال سازی عموما باعث بهتر شدن مدل می‌شود.

در ادامه هم نتایج پیش بینی کنار مقدار واقعی PM2.5 در بازه ۱۰ روزه برای مدل لگ یک روزه و لگ هفت روزه آورده شده است.



شکل ۷. مقدار پیش بینی شده PM2.5 برای لگ یک روزه



شکل ۸. مقدار پیش بینی شده PM2.5 برای لگ هفت روزه

همانطور که در شکل مشاهده می‌کنید برای مدل لگ هفت روزه مقدار پیش بینی شده تقریبا به مقدار واقعی فیت می‌شود.. اما برای مدل لگ یک روزه اینطور نیست و فقط روند را پیش بینی می‌کند. یعنی بالا پایین رفتنش را تونسته یاد بگیرد و مقدار پیش بینی شده کمی شیفت به بالا دارد.

در مقاله هم مدل لگ ۷ روزه نمایش داده شده است که شبیه شکل ماست البته دقت اون کمی بیشتر است.

مقدار r2, MAE, RMSE:

از کتابخونه sklearn ماژول metrics استفاده کردیم.

برای مدل با لگ یک روزه:

MAE: 96.966034
RMSE: 115.6379
R-Squared: 1.8181432891084746

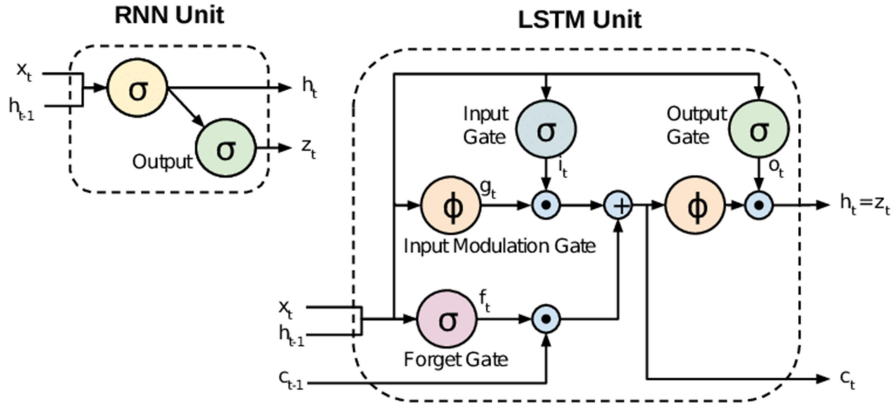
برای مدل با لگ هفت روزه:

MAE: 37.14925
RMSE: 67.308914
R-Squared: 0.602952317113865

پاسخ ۲ – تشخیص اخبار جعلی

۲-۱. توضیحات مدل‌ها

قسمت ۱. تفاوت RNN با LSTM: تفاوت اصلی آن‌ها در امکان یادگیری وابستگی بلند مدت است که توسط LSTM یادگرفته می‌شود اما در RNN این ظرفیت وجود ندارد و تنها تعداد محدودی از وابستگی‌ها توسط RNN قابل یادگیری هستند. دلیلش هم این است که در RNN ما مشکل محو شدگی گرادیان داریم یعنی وابستگی لایه مخفی فعلی به لایه‌های مخفی دورتر به خاطر چند مشتق پشت سرهم عملا صفر است! اما در LSTM واحدها حافظه داریم که می‌توانند اطلاعات رو برای زمان طولانی حفظ کنند. مشکل محوشدگی گرادیان و انفجار گرادیان با گیت‌هایinput و forget حل می‌شود چرا که اجازه می‌دهند بر جریان گرادیان کنترل بیشتری داشته باشیم و بتوانیم وابستگی بلند مدت را بهتر حفظ کنیم. که این باعث می‌شود کارایی LSTM بهتر از RNN باشد و هرچه قدر این وابستگی بلند مدت بیشتر اهمیت داشته باشد این تفاوت عملکرد هم بیشتر می‌شود. از آنطرف این افزونگی‌های LSTM نسبت به RNN نیاز به محاسبات بیشتری دارد که باعث می‌شود آموزش LSTM کندتر از RNN باشد.



شکل ۹. معماری **LSTM** و **RNN**

قسمت ۲. موثر بودن شبکه‌های بازگشتی برای داده‌های متنی: در اکثر موارد برای تحلیل و درک داده‌های متنی نیاز به نوعی به خاطر سپردن داریم مثلا برای پیش بینی کلمه بعدی باید کلمات قبل رو داشته باشیم یا وقتی می‌خواهیم موضوع یک بحث را بدانیم باید دنباله کلمات رو با هم برسی کنیم که شبکه‌های بازگشتی برای این مسائل مناسب است چون برای داده‌های توالی دار طراحی شده است و می‌تواند ارتباط مرحله فعلی را با مرحله قبلی یاد بگیرد. البته در کاربردهایی مثل classification که

برخی کلمات کلیدی تعیین کننده هستند CNNها بهتر عمل می‌کنند چون توانایی بیشتری برای استخراج ویژگی‌های محلی دارند.

قسمت ۳. شرح مدل ترکیبی مقاله: این مدل با توجه به توضیحات جواب قسمت ۲ برای بهره گیری از ظرفیت CNN در استخراج ویژگی‌های محلی و ظرفیت شبکه LSTM در یادگیری وابستگی بلند مدت از هر دو استفاده کرده و در مدل ارائه شده آن‌ها را ترکیب می‌کند.

ابتدا از یک لایه embedding برای تعبیه سازی استفاده می‌کند. سپس آن را به لایه کانولوشن یک بعدی می‌دهد تا ویژگی‌های محلی را استخراج کند. سپس از لایه MaxPooling برای انتخاب ویژگی‌ها و کاهش تعداد آن‌ها استفاده شده. خروجی آن را به LSTM داده که خروجی آن وابستگی بلند مدت بین ویژگی‌ها است. و در آخر از یک شبکه عصبی فولی کانکتت برای طبقه بندی اخبار به درست یا جعلی استفاه می‌کند.

تفاوت شبکه‌های بازگشتی با مدل ترکیبی در این است که مدل ترکیبی علاوه بر یادگیری وابستگی‌های بلند مدت می‌تواند به واسطه CNN قبل از آن یک استخراج ویژگی داشته باشد که باعث عملکرد بهتر آن شود.

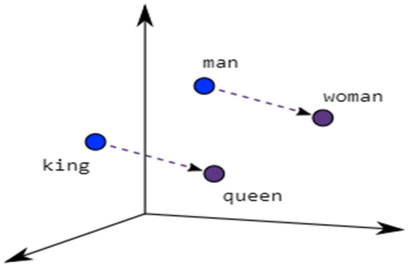
۲-۲. ورودی مدل‌ها

یکی از راه‌های اینکه متون را به مدل‌های شبکه عصبی بدهیم استفاده از Bag of word(BOW) است که از کلمات به عنوان feature برای ورودی مدل استفاده می‌کند. یعنی BOW مشخص می‌کند که هر کلمه چند بار در متن ظاهر شده است.که این برای هر داکیومنت یک بردار word-count می‌دهد که می‌تواند باینری هم باشد اگر کلمه‌ای از واژگان در داکیومنت ظاهر شده باشد یک می‌گذارد در غیر این صورت صفر می‌گذارد. سایز این وکتور برابر تعداد کلمات درون واژگان می‌باشد. این باعث شده که وکتور ما اسپارس باشد چون در داکیومنت خیلی از کلمات واژگان ظاهر نشده است. که این یعنی ورودی ما بسیار بزرگ است. و این باعث می‌شود تعداد پارامترهای شبکه مثل تعداد وزن‌ها زیاد شود و محاسبات سنگینی مورد نیاز باشد. همچنین بردار BOW به ما اطلاعات کمی می‌دهد چون توالی ظاهر شدن کلمات را در نظر نمی‌گیرد.

به دلیل مشکلاتی که گفته شده از word embedding برای داده‌های متنی استفاده می‌شود. که در این روش هر کلمه در دامنه و زبان مشخص به یک وکتور با مقادیر واقعی در فضای با ابعاد کمتر بازنمایی می‌شود.

مشکل اسپارس بودن در BOW با وکتور با ابعاد کم حل در word embedding حل می‌شود.

مشکل کمبود اطلاعات در BOW هم با قرار دادن بردارهای آیتم‌های مشابه کنار هم حل می‌شود. یعنی کلمات مشابه از نظر معنایی فاصله مشابهی در فضای بردار دارند. مانند شکل زیر که فاصله کلمه مرد از زن برابر فاصله شاه از ملکه است.



شکل ۱۰. فاصله کلمات مشابه در word embedding

انواع روش‌ها:

- Word2Vec: توسط گوگل ارائه شده است. که بر اساس فرضیه distributional hypothesis ارائه شده است.که می‌گوید کلمات مشابه در همسایگی هم قرار می‌گیرند. که ار دو روش CBOW و skip grams استفاده می‌کند.

۲-۳. پیاده سازی

۲-۳-۱. پیش پردازش

ابتدا دیتا را با دستور read_csv کتابخانه pandas می‌خوانیم.

سپس با دستور df.isna().sum() چک می‌کنیم که چندتا داده null است تا درصورت نیاز آن‌ها را با مقدار مناسب پر کنیم. که البته خوشبختانه همه خانه‌ها مقدار دارند.

سپس در محتوای مقالات می‌گردیم که آیا متن تکراری داریم یا خیر و آن‌ها را حذف کنیم. برای این کار از دستور زیر استفاده می‌کنیم:

df[‘article_content’].value_counts()[df[‘article_content’].value_counts() >1]

که مشاهده می‌کنیم ۱۵ تا تکراری داریم. که آن‌ها را با دستور drop_duplicates روی ستون ‘article_content’ جذف می‌کنیم.

حال برای تمییز سازی متون باید چند مرحله پیش پردازش کرد. که هر تابع یک وظیفه دارد:

تابع ()lowerize: کوچک کردن تمام حروف. چون تفاوتی ایجاد نمی‌کند حرفی بزرگ یا کوچک باشد و برای پردازش کمتر این کار رو می‌کنیم.

تابع ()remove_html: حذف ساختار html. هشتگ‌ها فایده‌ای برای ما ندارند.

تابع ()remove_url: حذف لینک‌ها. در تحلیل از آن‌ها استفاده نمی‌کنیم.

تابع ()remove_hashtags: حذف هشتگ‌ها. برای سادگی آن‌ها رو هم حذف می‌کنیم.

تابع ()remove_a: حذف کاراکتر @ از متن. اینکه این متعلق به چه کسی است در اینجا فایده‌ای برای ما ندارد.

تابع ()remove_brackets: حذف براکت.

تابع ()remove_stop_punct: حذف علائم نگارشی. یک لیستی از علائم نگارشی تهیه کردیم و آن‌ها رو حذف کردیم.

در تابع ()preprocessing همه این‌ها رو استفاده کردیم. بعد این تابع را با progress_apply روی ستون 'article' اعمال کردیم.

حال در کلاس ()TextClassificationDataset عملیات‌های توکنایز کردن دیتا با کتابخونه nltk، تبدیل کردن کلمات به لیستی از ایندکس‌ها، تبدیل این لیست به longTensor و در نهایت بریدن داده‌های اضافی از لیست(لیست بزرگتر از ۳۰۰). اگر هم کوچک‌تر از سایز ۳۰۰ باشند با پدینگ سایز آن را ۱۰۰ می‌کنیم.

۲-۳-۲. آموزش مدل‌ها

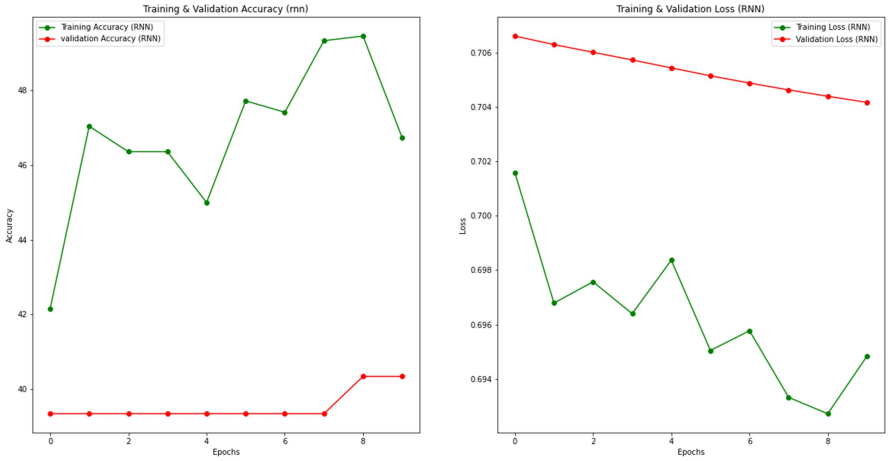
خب ابتدا از gensim برای بارگذاری GloVe با ابعاد ۱۰۰ استفاده کردیم. که بعدا در مدل از آن استفاده کنیم. که این مدل از روی دیتاست ویکی پدیا pre-train شده است.

داده‌ها را با تابع get_splits به سه قسمت train,val, test با نسبت 0.8,0.1,0.1 به صورت تصادفی تقسیم می‌شوند. و سپس به دیتالودر با بچ سایز ۶۴ برای train و ۴ برای test ,val داده می‌شوند. اکنون داده آمده دادن به مدل است.

مدل پایه ما یک LSTM با دو لایه است. که به سادگی با پایتورچ در کلاس LSTMModel پیاده سازی کردیم. به اینصورت که ابتدا در لایه اول embedding رو اعمال کردیم. سپس از Lstm استک شده استفاده کرده و خروجی آن‌ها رو به هم وصل می‌کنیم. در آخر هم یک لایه فولی کانکتت استفاده کردیم.

```
LSTMModel(
  (embeddings): Embedding(2844, 100, padding_idx=0)
  (lstm): LSTM(100, 16, num_layers=2, batch_first=True)
  (linear): Linear(in_features=32, out_features=1, bias=True)
  (sig): Sigmoid()
  (relu): ReLU(inplace=True)
)
```

۱۰ ایپاک آن را آموزش دادیم و نتایج زیر برای loss و acc آن بدست آمد.



شکل ۱۱. نمودار loss و accuracy برای مدل پایه

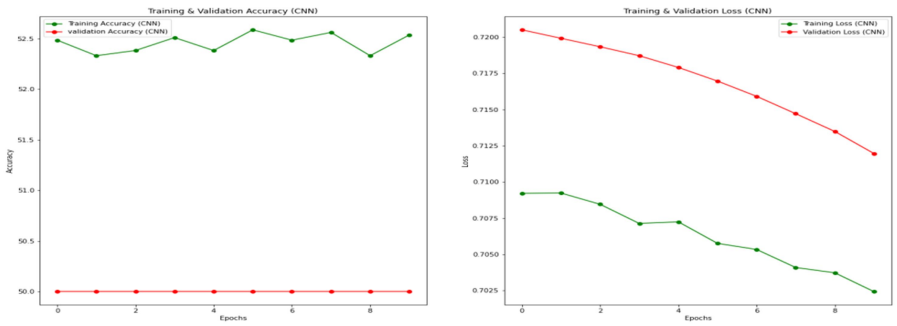
همچنین از کتابخانه sklearn.metics و ماژول classification_report آن برای گزارش معیارهای خواسته شده استفاده کردیم که نتایج آن به صورت زیر است:

	precision	recall	f1-score	support
Predicted Fake	0.52	0.98	0.68	41
Predicted True	0.67	0.05	0.10	39
accuracy			0.53	80
macro avg	0.59	0.51	0.39	80
weighted avg	0.59	0.53	0.39	

پس از آن مدل ترکیبی مقاله را پیاده سازی کردیم. که در آن بعد از embedding ،یک لایه کانولوشنال برای استخراج ویژگی(n-gram را در نظر بگیرید)میاد سپس یک لایه maxPooling برای انتخاب فیچرها و جلوگیری از بیش برازش استفاده شده. و در ادامه از یک لایه LSTM استفاده شده که از یادگیری ارتباط کلمات کنار هم برای پیش بینی بهتر کمک بگیرد. در آخر از یک لایه فولی کانکتت برای دسته بندی اخبار استفاده کردیم. که از تابع فعال ساز سیگموید هم می‌گذرد که بتواند طبقه بندی باینری را درست انجام دهد.

Hybrid_CNN_LSTM((embeddings): Embedding(2844, 100, padding_idx=0) (conv1d): Conv1d(300, 128, kernel_size=(5,), stride=(1,)) (dropout): Dropout(p=0.2, inplace=False) (linear): Linear(in_features=32, out_features=1, bias=True) (maxpooling1d): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (lstm): LSTM(48, 32, batch_first=True))

این مدل را هم ۱۰ ایپاک آموزش دادیم. که نمودار loss و accuracy آن در زیر آورده شده:



شکل ۱۲. نمودار loss و accuracy برای مدل ترکیبی مقاله

همچنین دوباره از کتابخانه sklearn.metics و ماژول classification_report آن برای گزارش معیارهای خواسته شده استفاده کردیم که نتایج آن به صورت زیر است:

	precision	recall	f1-score	support
Predicted Fake	0.00	0.00	0.00	32
Predicted True	0.60	1.00	0.75	48
accuracy			0.60	80
macro avg	0.30	0.50	0.37	80
weighted avg	0.36	0.60	0.45	80

مقایسه: روش مدل ترکیبی دقت ۶۰ درصد داده است. اما روش مدل پایه دقت ۵۳ درصد داده است.

اما اگه به سایر معیارها هم نگاه کنیم میفهمیم که انگار آنقدر هم بهتر نیست. چون به نظر میرسه هر دو مدل علاقه دارند که تمام اخبار را یا منفی یا مثبت پیش بینی کنند و این بد است.

نمودار loss رو اگه مقایسه کنیم هر دو کاهش داشتند اما برای مدل ترکیبی این کاهش بهتر و سرراست است.

نمودار accuracy برای train های هر دو مدل خوب بوده اما برای validation یا ثابت بوده یا تقریبا ثابت بوده است.

۲-۴. تحلیل نتایج

به نظر یکجای کار مشکل داره که اکثر دیتاها فیک طبقه بندی می‌شوند. اگه با هایپر پارامترها هم بازی کنیم تنها تغییر حاصله این است که اینبار تقریبا همه دیتاها درست تشخیص داده می‌شوند.

این مشکل میتونه چند دلیل داشته باشه:

۱. کمبود دیتا: می‌دانیم که شبکه‌های عمیق به دیتا زیاد نیاز دارند تا اورفیت نشوند و درست کار کنند. اما در اینجا دیتاست ما فقط حاوی ۷۸۹ دیتا است. که ۲۰ درصد آن هم صرف دیتا تست و ولیدیشن می‌شود. برای همین طبیعی است که مدل overfit شود.

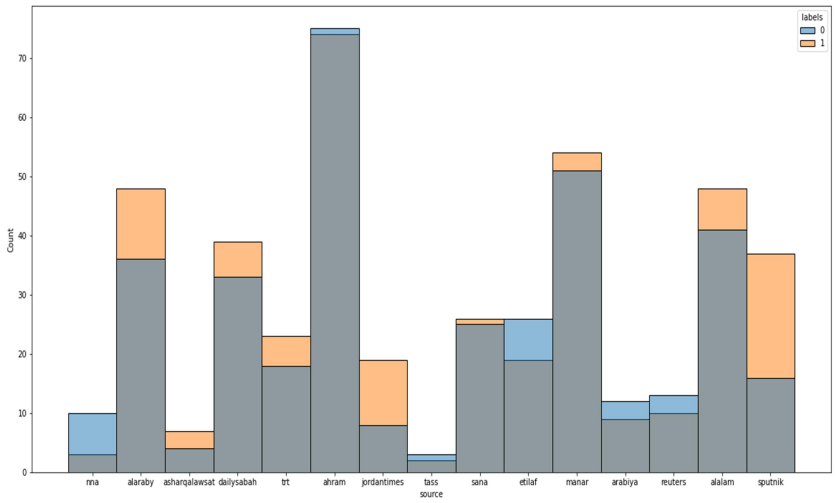
۲. هایپر پارامترها: یکی از دلایل این مشکل می‌تواند تنظیم نبودن مناسب مقادیر آن‌ها باشد. که البته چندتایی رو امتحان کردم اما تغییر محسوسی ایجاد نشد.

۳. شاید هم مشکل از مدل است. هرچند که بعید است اما شاید مدل دقیقا شبیه مقاله پیاده نشده باشد. چون در مقاله مثلا برای سایز embedding و حداکثر طول خبر متناقص صحبت کرده. در متن گفته که سایز embeddingکه pre train شده است ۱۰۰ است و حداکثر دنباله ۳۰۰ است. اما در شکل‌های که کشیده برعکس عمل کرده است. یا شاید تابع فعال سازهای اشتباهی استفاده شده. یا مثلا جایی باید ابعاد را تعویض می‌کردم که نکردم.

در کل هم شبکه ترکیبی بهتر از شبکه LSTM عمل می‌کنه.

.احتمالا اگه دیتاست دیگر را انتخاب می‌کردیم به دلیل بزرگتر بودنش دقت بهتری بدست می‌آید. اگر دقت کنید نمودار loss دیتای train خیلی بهتر از دیتایvalid کم میشه.شاید این دلیلش اینکه مدل ما قدرت تعمیم سازی خوبی نداره و برای همین احتمالا استفاده از dropout یا شاید حتی maxPooling این مشکل را تا حدی برطرف کند.

یا اینکه از دیتاهای دیگر مثل تیتر خبر، انتشار دهنده خبر استفاده کرد. مثلا نمودار زیر نشان میده که با منبع خبر میشه تشخیص داد خبر احتمالا فیک است یا نه:



شکل ۱۳. هیستوگرام منابع خبری با دسته بندی

.