به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

پـزدازش زبـان هـای طبـیـعـی

CA4

فاطمه سلیقه

۸۱۰۱۹۸۳۰۶

فروردین ماه ۱۳۹۹

RNN

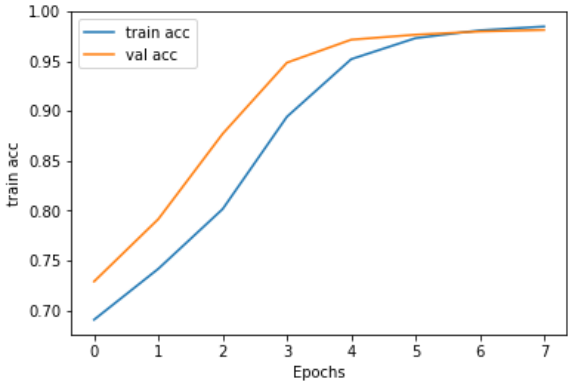
ابتدا فایل RNN_train و RNN_test را خوانده و جمله ها را در یک لیست و سپس POS ها را در یک لیست دیگر قرار می دهیم پس از آن به جای هر کلمه index آن کلمه را قرار می دهیمو در واقع مدل را کد می کنیم سپس از pad_sequence استفاده می نماییم تا داده های ورودی به وکتورهایی با اندازه ۷۰ شوند . زیرا بزرگترین اندازه یک جمله ۷۰ است به همین صورت داده های تست را هم با اندازه ۷۰ در نظر می گیریم . سپس با استفاده از to_categorical خروجی های هر دو داده تست و آموزش را one hot می کنیم .

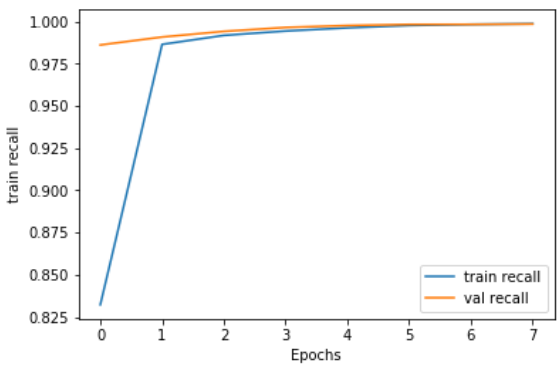
برای داده های تست هم به مانند داده های آموزش وکتورهای کدشده ایجاد می کنیم و برای کلماتی که در داده های تست قرار دارند ولی در آموزش نیستند از به جای آنها از - unk - استفاده می کنیم . در واقع در هنگام تبدیل کلمات به کدها یک کلمه به نام – unk - و یک tag به نام – unk - اضافه نموده ایم .

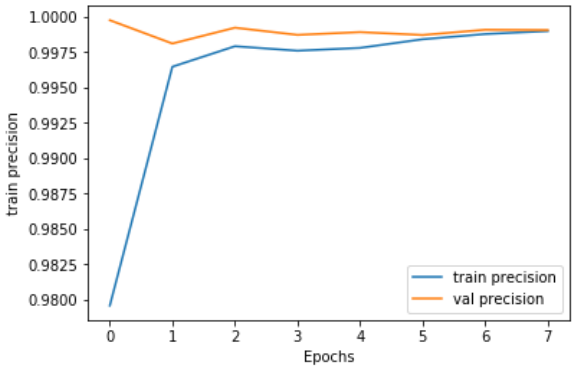
حال مدل را پیاده سازی نموده که از یک لایه Embedding و یک لایه LSTM و یک لایه پنهان و یک لایه خروجی تشکیل شده است . که به صورت زیر است :

```
model = Sequential()
model.add(InputLayer(input_shape=(maxLen, )))
model.add(Embedding(len(wordCoding), 128))
model.add(Bidirectional(LSTM(256, return_sequences=True,dropout=0.1, recurrent_dropout=0.1)))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(42, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer=Adam(0.002),metrics=[keras_metrics.precision(), keras_metrics.recall(
```

در ابتدا از لایه پنهان استفاده ننموده و دقت 0.7 داشت سپس یک لایه پنهان اضافه نموده با ۶۴ نورون و دقت به حدود 0.9 رسید با افزایش تعداد نورون ها به 128 نتایج زیر به دست آمد :







مقادیر به دست آمده برای داده های تست :

```
loss =  0.16097455946207045 acc =  0.9786485710144043  precision =  0.9926917402027482 recall =  0.9989406259710251
```

و در نهایت خروجی را در فایل outputRNN.xlsx ذخیره شده است .

HMM

برای پیاده سازی HMM لازم است تا ماتریس های emission probability و transiotion probability را تولید کنیم . برای محاسبه emission probability محاسبه می کنیم که برای هر tag یک کلمه چندبار تکرار شده است سپس احتمال را محاسبه می کنیم . برای هر tag مجموع احتمالات برابر یک است . سپس برای محاسبه transition probability از bigrams استفاده می کنیم . در واقع تمام tagها را در یک لیت قرار داده و آن ها را دو تا دو تا در نظر گرفته و احتمال هر دوتا را در نظر می گیریم . سپس تابع Viterbiرا پیاده سازی نموده و با استفاده از تابع Viterbi الگوریتم Viterbi را اجرا می کنیم . به اینصورت که یک ماتریس ایجاد می کنیم که سطرهای آن tagها هستند و ستون های آن کلمات . در واقع برای هر جمله یک ماتریس ایجاد می کنیم . مشکلاتی که وجود داشت این بود اول اینکه برخی کلمات در داده تست بودند ولی در داده آموزش نبودند و یا احتمال انتقال از یک tag به tag دیگر وجود نداشت برای حل این مشکل یک احتمال کم برای این موارد درنظر می گیریم مثلا ۰,۰۰۰۰۰۰۰۰۱ . در نهایت پس از پایان محاسبه ماتریس Viterbi عمل backtracking را انجام داده و tag ها را پشت هم می نویسیم .

نتایج به دست آمده از تست به صورت زیر است :

```
accuracy =  0.8871004001618632

Precision =  0.9274104498228184

recall =  0.8871004001618632
```

و در نهایت خروجی در فایل outputHMM.xlsx قرار داده شده.

با توجه به نتایج به دست آمده اگر چه به دلیل اینکه داده های تست وآموزش در دو مدل بالا یکسان نبودند شاید نتوان به صورت دقیق مقایسه کرد اما به طور کلی مدل RNN بهتر از مدل HMM عمل می کند .