تمرین سوم کامپیوتری

(ثمین مهدی زاده – ۸۱۰۱۰۰۵۲۶)

در این تمرین به کمک دیتاست penn treebank مدل هایی برای پیش بینی نقش کلمات و گروه های اسمی آموزش داده شده اند. در ادامه هر یک از مدل ها با یکدیگر مقایسه شده و نتایج حاصل از آن ها به دست آمد.

تعیین نقش کلمات

الف)

در ابتدا جملات را به صورت عادی و سپس با tag-set='universal' لود کرده و نتایج زیر بر روی اولین جمله دیتاست به صورت زیر است:

```
without universal
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'), (',', ','), ('wil
l', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('dir
ector', 'NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]
tags:  {'-RRB-', 'VBN', 'JJS', '-LRB-', 'MD', 'VBD', 'RBR', 'SYM', 'WDT', '.', ',', 'VBP', 'CC', "''", '-NONE-', 'I
N', ':', '#', 'PRP$', 'RB', 'WP', 'RP', 'NNP', 'PDT', 'JJR', 'WP$', 'LS', 'NN', 'POS', 'JJ', '``', '$', 'CD', 'NNS'
, 'UH', 'VBZ', 'WRB', 'VB', 'FW', 'RBS', 'DT', 'VBG', 'EX', 'NNPS', 'TO', 'PRP'}
tag lenght:  46
universal
[('Pierre', 'NOUN'), ('Vinken', 'NOUN'), (',', '.'), ('61', 'NUM'), ('years', 'NOUN'), ('old', 'ADJ'), (',', '.'),
('will', 'VERB'), ('join', 'VERB'), ('the', 'DET'), ('board', 'NOUN'), ('as', 'ADP'), ('a', 'DET'), ('nonexecutive'
, 'ADJ'), ('director', 'NOUN'), ('Nov.', 'NOUN'), ('29', 'NUM'), ('.', '.')]
tags:  {'.', 'NUM', 'DET', 'CONJ', 'ADJ', 'NOUN', 'PRON', 'X', 'ADP', 'ADV', 'VERB', 'PRT'}
tag lenght:  12
```

مشاهده می شود تعداد تگ ها در حالت عادی بیشتر بوده و شامل جزئیات بیشتری است. برای مثال مشاهده می شود که در حالت universal برای تمام کلمات pierre,vinken,years,board تگ NOUN به کار رفته در حالتی که در حالت عادی هر کدام تگ های مجزا دارند.

ب)

برای ارزیابی مدل و تنظیم هایپر پارامتر ها لازم است تا داده ها به سه دسته train,test,split تقسیم کنیم.به این منظور ۱۵ درصد داده ها برای تست، حدود ۱۲ درصد برای validation و مابقی برای train در نظر گرفته شده اند.(۷۳ درصد)

```
train_set, test_set = train_test_split(dataset, test_size=0.15, random_state=42)
train_set, valid_set = train_test_split(train_set, test_size=0.15, random_state=42)
```

لازم به ذکر است که در پیاده سازی الگوریتم Viterbi چون هایپر پارامتر نداشتیم داده ها تنها با نسبت ۱۵ به ۸۵ تقسیم شده اند.

پ)

الگوریتم Viterbi به این صورت عمل می کند که پس از ساختن ماتریس های emission(احتمال وقوع کلمه)و transition(احتمال رفتن از یک تگ به تگ بعدی) در هر لحظه به کمک احتمالات به دست آمده از آخرین کلمه دیده شده احتمال کلمه بعد را به گونه ای انتخاب می کند که بیشینه شود. در واقع این احتمال به صورت زیر محاسبه می شود:

Max(P_emission(word)*p_transtion(tag2|tag1)*p_prev(word))

به طوری که p_prev(word) احتمال وقوع دادن دنباله تا word را در تگ های مختلف در بر دارد.
سودوکد این الگوریتم در شکل زیر قابل مشاهده است.

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi*$[N+2,T]$
for each state s **from** 1 **to** N **do** ;initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ;recursion step
for each state s **from** 1 **to** N **do**
$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Figure 5.17 Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state-path through the HMM which assigns maximum likelihood to the observation sequence. Note that states 0 and q_F are non-emitting.

پس از آموزش مدل به کمک داده های train و به دست آوردن ماتریس های emission و transition دقت مدل بر روی داده های تست برابر ۹۰ درصد بود که نشان می دهد الگوریتم عملکرد خوبی داشته است.

ت)
تصویر زیر تعدادی از جملات که برخی کلمات آن به اشتباه تشخیص داده شده اند را نشان می دهد:

```
viterbi.print_wrong_tags(10)
```

```
sent For the Agency for International Development , appropriators approved $ 200 million *U* in secondary loan guar
antees under an expanded trade credit insurance program , and total loan guarantees for the Overseas Private Invest
ment Corp. are increased *-3 by $ 40 million *U* over fiscal 1989 as part of the same Poland package .
w_word ['appropriators', 'expanded', 'Overseas', 'Private', '1989']
w_real ['NOUN', 'VERB', 'NOUN', 'NOUN', 'NUM']
w_pred ['PRON', 'ADJ', 'ADJ', 'ADJ', 'NOUN']
--------------------------------------------------
sent The market is just becoming more efficient . ''
w_word ['more', 'efficient']
w_real ['ADV', 'ADJ']
w_pred ['ADJ', 'NOUN']
--------------------------------------------------
sent Moscow has settled pre-1917 debts with other countries in recent years at less than face value .
w_word ['pre-1917']
w_real ['ADJ']
w_pred ['DET']
--------------------------------------------------
sent `` Unemployment continues at a relatively low level , * providing a sense of job security , and a low inflatio
n rate has kept the purchasing power of the weekly paycheck reasonably strong . ''
w_word ['Unemployment', 'purchasing', 'weekly', 'paycheck', 'reasonably']
w_real ['NOUN', 'VERB', 'ADJ', 'NOUN', 'ADV']
w_pred ['PRON', 'NOUN', 'NOUN', 'ADP', 'DET']
--------------------------------------------------
sent At the same time , though , he chastised the media for * paying such close attention to Japanese investment wh
en other foreign countries , notably Britain , are acquiring more American assets *T*-1 .
w_word ['though', 'close', 'notably', 'acquiring', 'American']
w_real ['ADV', 'ADJ', 'ADV', 'VERB', 'ADJ']
w_pred ['ADP', 'NOUN', 'DET', 'DET', 'NOUN']
--------------------------------------------------
sent *-1 Funded *-2 by a $ 1 million *U* gift from Tokio Marine & Fire Insurance , the service will follow Japanese
medical protocols , including emphasis on preventative medicine .
w_word ['gift', 'Tokio', 'follow', 'emphasis', 'preventative']
w_real ['NOUN', 'NOUN', 'VERB', 'NOUN', 'ADJ']
w_pred ['VERB', 'DET', 'DET', 'X', 'DET']
```

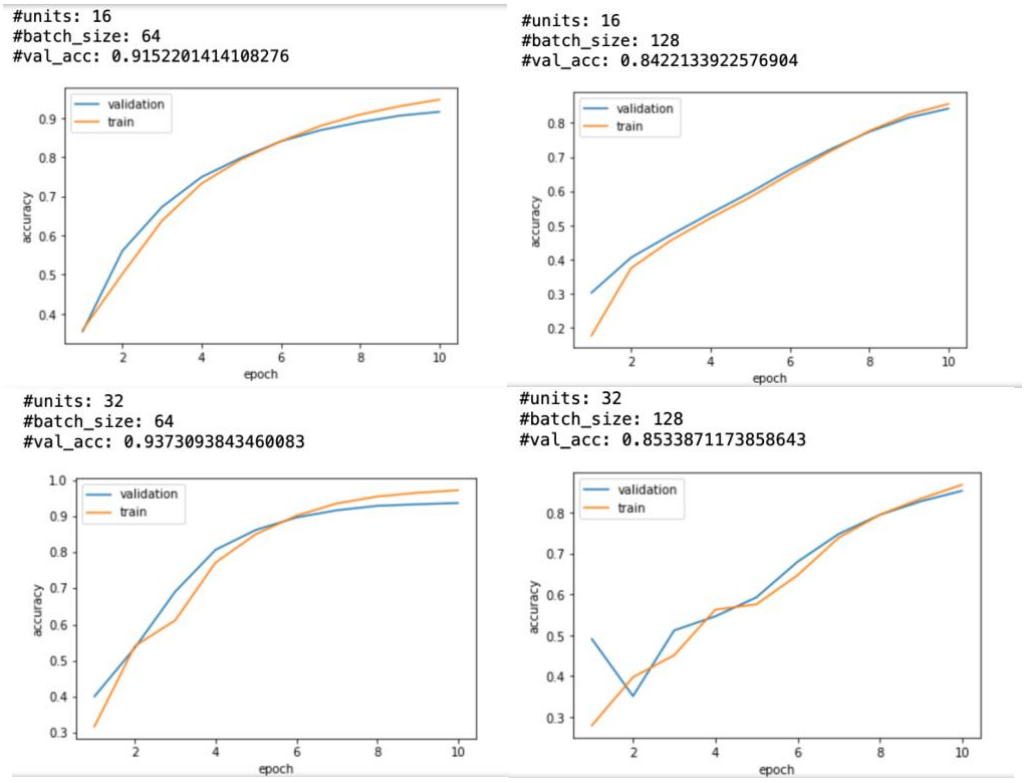
اگر به جمله اول نگاه کنیم می بینیم کلمه مانندprivate به اشتباه دسته بندی شده است. البته می توان گفت این اشتباه خیلی غیر محتمل نیست چرا که این کلمه بسته به متنی که در آن قرار دارد هم می تواند به عنوان NOUN و هم می تواند به عنوان ADJ باشد. در این جا ممکن است به دلیل بیشتر بودن private در نقش ADJ در داده های آموزش مدل تصمیم اشتباهی گرفته باشد. مورد مشابه را می توان در جمله ی آخر و برای کلمه gift دید که می تواند در نقش فعل و یا اسم باشد که مدل دوباره اشتباه تشخیص داده است. علاوه بر این کلمه ی pre-1917 نیز کلمه ی پر تکراری نیست و احتمالا خیلی کم و یا اصلا در داده های آموزش نبوده است و به همین علت مدل در تشخیص آن ضعیف عمل کرده است و ممکن است تصمیم را بیشتر بر این اساس گرفته باشد که بعد از فعل وقوع کدام یک از این تگ ها محتمل تر است.

ث)

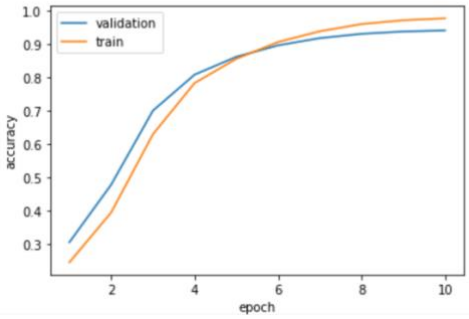
برای برخورد با کلمات ناشناخته از smoothing استفاده شد تا صفر شدن احتمال مربوط به یک کلمه کل احتمال جمله را صفر در نظر نگیرد و همچنان به آن شانسی برا انتخاب شدن بدهد. لازم به ذکر است بهترین راه برای جلوگیری از این نوع خطاها جمع آوری داده های بزرگ است که تا حد ممکن امکان مشاهده کلمات جدید را کاهش دهد. راه دیگر برای برخورد با کلمات ناشناخته استفاده از قواعد زبان برای تخصیص تگ است برای مثال میدانیم در زبان انگلیسی همیشه فعل بعد از فاعل می آید و در صورتی که فاعل را مشاهده نکردیم نباید به کلمه ای حتی ناشناخته نقش فعل را بدهیم. علاوه بر این می توانیم از احتمالات نیز استفاده کرده برای مثال می توانیم یک نوع تگ خاص بیشتر با صفت همراه می شود و در این صورت به کلمه ناشناخته نقش مربوطه را اختصاص دهیم.

ج)

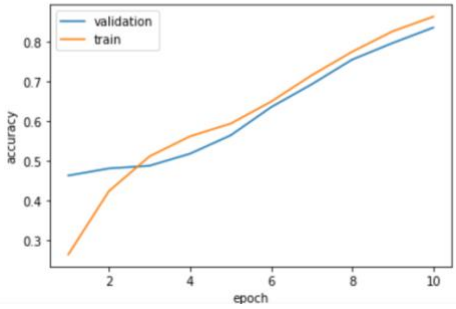
هایپر پارامتر های در نظر گرفته شده برای این قسمت سایز hidden-layer و batch بوده اند که با استفاده از داده ی validation بهترین آنها انتخاب شد. برای این قسمت ۶ مدل مختلف امتحان شد که نتایج آن به شرح زیر است:



#units: 64
#batch_size: 64
#val_acc: 0.9420745372772217



#units: 64
#batch_size: 128
#val_acc: 0.8346985578536987



مطابق تصاویر بالا بهترین مدل انتخاب شده برای RNN مدلی با سایر hidden_layer برابر ۶۴ و batch مساوی ۶۴ است.

مشاهده می شود که هنگامی که سایز batch را برابر ۶۴ گرفتیم هرچه تعداد unit ها بیشتر باشد دقت مدل بالاتر است اما لزوما نمی توان گفت که افزودن تعداد unit ها موثر است و دقت را افزایش دهد چرا که ممکن است این افزودن پیچیدگی مدل را زیاد کند و باعث overfit شدن آن شود. همان طور که می بینیم هنگامی که از تعداد batch برابر ۱۲۸ استفاده کردیم افزودن تعداد unit ها از ۳۲ به ۶۴ دقت را برای داده هایvalidation کاهش داده است.

علت استفاده از داده ی validation تنظیم کردن هایپر پارامتر های مدل است. در واقع در این جا سعی داریم بدون مشاهده داده ی تست بهترین مدل را انتخاب کنیم و سپس نتایج را بر روی داده های تست امتحان کنیم، در صورتی که برای هایپرپارامتر ها از داده ی تست استفاده کنیم در واقع انگار تنها مدل را برای خوب جواب دادن بر روی داده تست مورد نظر آموزش داده ایم و مدل تعمیم یافته ای نداریم.

چ)

به کمک قسمت بالا بهترین پارامتر ها را انتخاب کرده و نتایج آن بر روی تست را به دست می آوریم:

(با توجه به این که برای آموزش مدل padding اضافه شده بود کسری از دقت مدل به درست حدس زدن padding ها بر میگردد. برای رفع این مشکل به کمک mask داده های padding را از بین برده و سپس دقت به دست می آید)

Layer (type)	Output Shape	Param #
embedding_245 (Embedding)	(None, 100, 150)	1500000
simple_rnn_136 (SimpleRNN)	(None, 100, 64)	13760
time_distributed_245 (TimeD istributed)	(None, 100, 13)	845

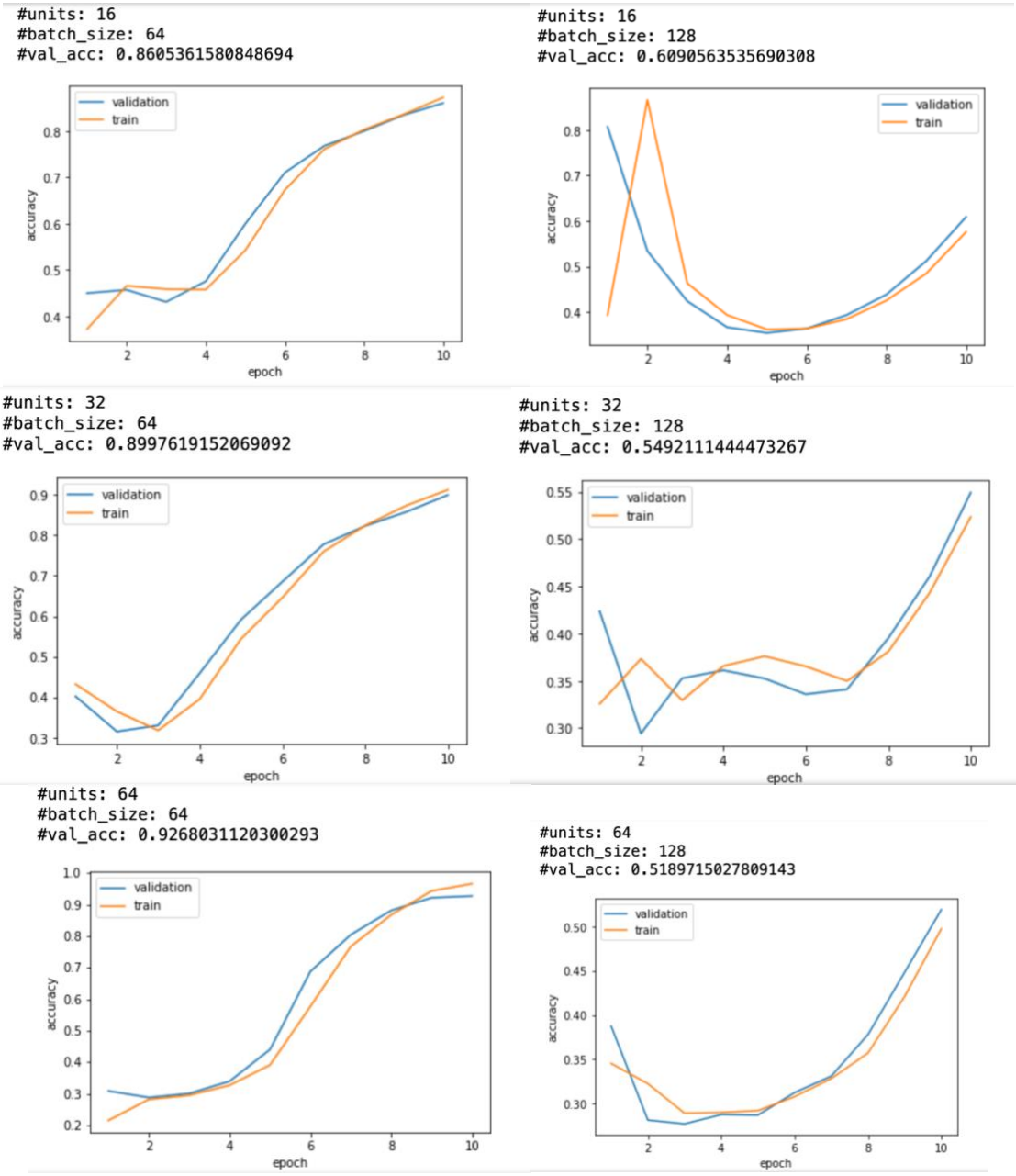
Total params: 1,514,605
Trainable params: 1,514,605
Non-trainable params: 0

دقت این مدل بر روی داده های تست به صورت زیر است:

19/19 [==============================] - 0s 9ms/step - loss: 0.0656 - ignore_accuracy: 0.9386
#accuracy: 0.9385961294174194

ح)

تعیین هایپر پارامتر ها برای LSTM



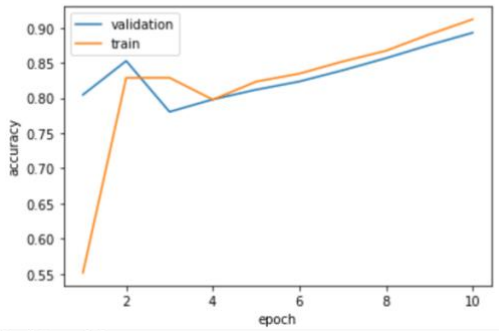
تصاویر بالا نشان می دهد تفاوت عملکرد مدل بر روی سایز بچ برابر با ۶۴ و یا ۱۲۸ بسیار متفاوت است. هنگامی که سایز بچ برابر با ۶۴ بوده مدل به دقت های بسیار بیشتری رسیده است علاوه بر این با افزایش تعداد unit ها دقت بالا رفته در خالی که در بچ سایر برابر با ۱۲۸ با زیاد کردن unit ها دقت مدل کم شده است.
بهترین مدل انتخابی مدلی با batch size=64 و unit = 64 است.

ارزیابی این مدل بر روی داده های تست نتایج زیر را می دهد:

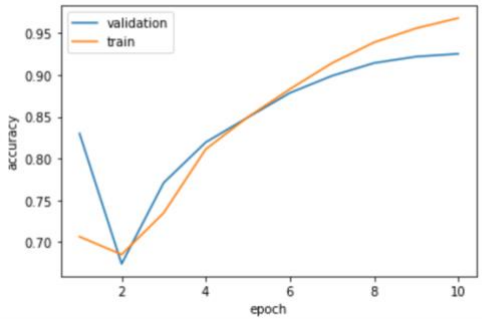
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_259 (Embedding)   (None, 100, 150)          1500000

 lstm_76 (LSTM)              (None, 100, 64)           55040

 time_distributed_259 (TimeD  (None, 100, 13)          845
 istributed)

=================================================================
Total params: 1,555,885
Trainable params: 1,555,885
Non-trainable params: 0
_________________________________________________________________
19/19 [==============================] - 0s 19ms/step - loss: 0.0708 - ignore_accuracy: 0.9363
#accuracy: 0.9363410472869873
```

تعیین هایپر پارامتر ها برای GRU

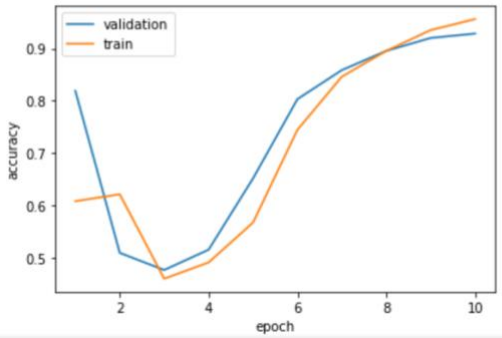
#units: 16
#batch_size: 128
#val_acc: 0.8926644325256348



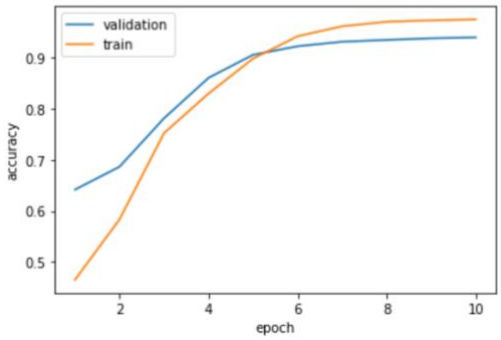
#units: 16
#batch_size: 64
#val_acc: 0.9254303574562073



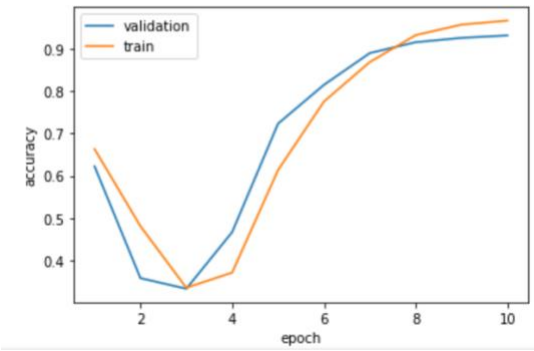
#units: 32
#batch_size: 128
#val_acc: 0.9282424449920654



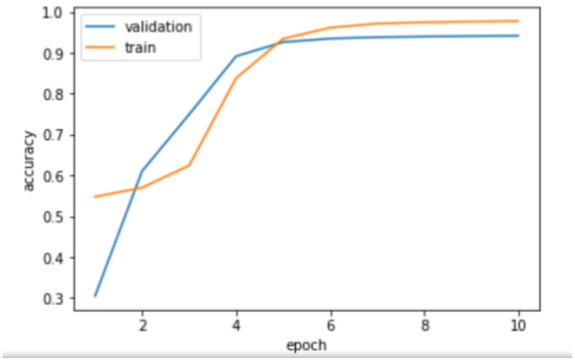
#units: 32
#batch_size: 64
#val_acc: 0.9388243556022644



```
#units: 64
#batch_size: 128
#val_acc: 0.9321743249893188
```




```
#units: 64
#batch_size: 64
#val_acc: 0.9416213035583496
```



برای این مدل در هر دو سایز batch با افزایش تعداد unit ها دقت مدل افزایش می یابد اما به طور کلی مدل در batch size=64 بهتر عمل کرده است.

بهترین مدل انتخاب شده مدلی با batch size=64 و unit = 64 است.

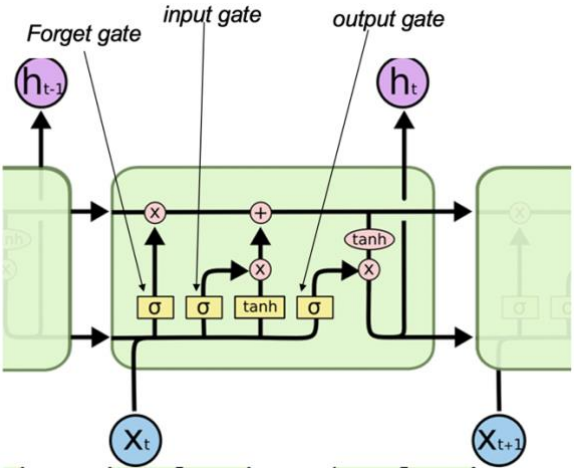
```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_252 (Embedding)    (None, 100, 150)          1500000

 gru_47 (GRU)                 (None, 100, 64)           41472

 time_distributed_252 (TimeD  (None, 100, 13)           845
 istributed)

=================================================================
Total params: 1,542,317
Trainable params: 1,542,317
Non-trainable params: 0
```

```
19/19 [==============================] - 0s 16ms/step - loss: 0.0501 - ignore_accuracy: 0.9420
#accuracy: 0.9419696927070618
```

دقت به دست آمده از مدل های مختلف از کمترین به بیشترین برابر GRU,RNN,LSTM است. LSTM از آن جا که گیت های بیشتری دارد انتظار میرفت که از بقیه مدل ها بهتر عمل کند البته رنج دقت ها بسیار نزدیک به هم هستند و می توان گفت احتمالا به دلیل تعداد epoch کم در نظر گرفته شده(۱۰) پارامتر های مدل LSTM به خوبی آموزش ندیده اند و در صورتی که تعداد epoch ها را بیشتر کنیم احتمالا در نهایت این مدل بیشترین دقت را به دست می آورد. مدل GRU نیز تقریبا شبیه به lSTM است با این تفاوت که تعداد گیت های کمتری دارد اما به هر حال از RNN قدرت بیشتری دارد و به همین علت است که دقت بالاتری دارد. در واقع مدل GRU داری دوگیت reset و update است که به آن اجازه می دهد در هر مرحله نسبت به RNN تصمیمات بهتری بگیرد.

خ)

گیت های LSTM در شکل زیر نشان داده شده است که در ادامه راجع به آنها توضیح داده می شود:



Forget

این گیت مشخص می کند که چه مقدار از اطلاعات cell state وارد شوند (در واقع به کمک تابع سیگموند مقادیر cell state قبلی در مقادیر ۰ و ۱ ضرب می شوند و به این ترتیب اهمیت آنها در تصمیم گیری مشخص می شود)

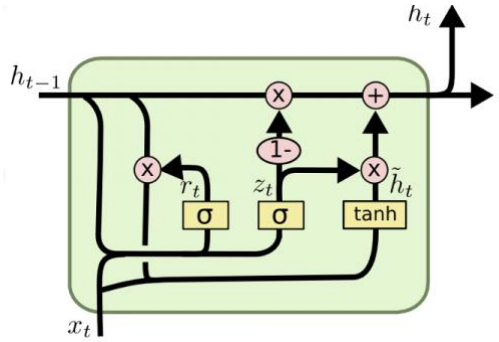
Input

این گیت مشخص می کند چه مقدار از اطلاعات جدید وارد cell state شود تا در ادامه از آنها استفاده شود.(ضریب اهمیت اطلاعات مجددا توسط سیگوید اعمال می شود)

Output

خروجی اطلاعات حاصل از cell state و اطلاعات جدید برای استفاده در زمان بعدی.

در مدل GRU دو گیت forget و input با یکدیگر ترکیب شده و update gate را به وجود می آورند. بنابراین در این مدل تنها دو گیت داریم. علاوه بر این cell state و hidden state نیز با یکدیگر ترکیب شده و تنها یک خط وجود دارد:



Update

این گیت دقیقا مانند دو گیت فراموشی (Forget Gate) و ورودی (Input Gate) در شبکه LSTM عمل می‌کند. این گیت تصمیم می‌گیرد چه مقدار از اطلاعات گذشته، یعنی اطلاعاتی که در گام‌های قبلی داشتیم، به شبکه اضافه شود. در این گیت مقدار ورودی جدید (x_t) به‌همراه مقدار حالت نهان گام قبلی (h_{t-1}) در وزن متناظر خود ضرب و سپس با هم جمع می‌شوند و به یک تابع سیگموید وارد می‌شوند تا خروجی میان بازه صفر تا ۱ قرار بگیرد. در زمان آموزش شبکه این وزن‌ها هر بار به‌روزرسانی می‌شوند تا فقط اطلاعات مفید به شبکه اضافه شوند.

Reset
این گیت تصمیم می‌گیرد چه مقدار از اطلاعات گذشته، یعنی اطلاعات گام‌های قبلی، فراموش شود. در اینجا هم مقدار ورودی جدید(x_t) ، به‌همراه مقدار حالت نهان گام قبلی(h_{t-1}) ، در وزن متناظر خود ضرب و سپس با هم جمع می‌شوند و به یک تابع سیگموید وارد می‌شوند تا خروجی بین بازه صفر تا ۱ قرار بگیرد. تفاوتی که با گیت به‌روزرسانی دارد این است که وزن‌هایی که مقدار ورودی و حالت نهان گام قبلی در آن ضرب می‌شوند متفاوت است و این یعنی بردارهای خروجی در اینجا با بردار خروجی که در گیت به‌روزرسانی داریم متفاوت خواهد بود.

همان طور که مشاهده می شود در GRU سعی شده با کمتر کردن پارامتر ها سرعت مدل بیشتر شود اگرچه LSTM به دلیل در نظر گرفتن اطلاعات بیشتر احتمالا یادگیری بیشتری دارد و به دقت های بالاتری می رسد.(البته بسته به دیتاست ممکن است هر کدام از این مدل ها دقت بهتری بدهند)

د)

بهترین نتیجه مربوط به GRU با دقت حدود ۹۴ بود در حالی که مدل Viterbi به دقت حدود ۹۰ رسیده بود. می توان گفت هر دو به مدل به دقت های خوبی رسیده اند اما مدل GRU به دلیل در نظر گرفتن اطلاعات بیشتر، استفاده از embedding کلمات و یا انتقال اطلاعات از زمان های قبلی به بعدی متناسب با ضرایب آموزش دیده شده طبیعتا بهتر عمل کرده است.

تعیین گروه های اسمی

الف)

در این بخش الگوریتم Viterbi را طوری تغییر داده که گروه های اسمی جملات را با در نظر گرفتن توالی های غیر مجاز مشخص کند. برای این کار از دیتاست penn tree و به کمک chunk تگ های مربوط به داده به دست آمدند.

ب)

در این بخش مجددا داده های تست و ترین به نسبت ۱۵ به ۸۵ تقسیم شده و آموزش صورت گرفته است. در این قسمت برای جلوگیری از توالی های غیر مجاز کافی است احتمال وقع آن را صفر کنیم تا الگوریتم Viterbi هیچ گاه مسیری که از این حالات میگذرد را انتخاب نکند. به این منظور پس از smooth کردن ماتریس transition به صورت دستی حالت های غیر مجاز برابر با صفر گذاشته می شود تا با ضرب شدن احتمالات مسیری که از این حالت عبور می کند همواره مقدار احتمال برای توالی برابر صفر در نظر گرفته شود:

```
row_tag = self.transition_df.index
row_tag = list(row_tag)
for tag1 in row_tag:
    for tag2 in self.tags:
        if((tag1=='O' and tag2[:2]=='I-')) :
              self.transition_df.loc[tag1,tag2] = 0
        if(tag1[:2]=='I-' and tag2[:2]=='I-' and tag1[2:]!=tag2[2:]):
              self.transition_df.loc[tag1,tag2] = 0
        if(tag1[:2]=='B-' and tag2[:2]=='I-' and tag1[2:]!=tag2[2:]):
            self.transition_df.loc[tag1,tag2] = 0
return self.transition_df
```

پ)

عملکرد مدل به صورت زیر است:

	precision	recall	f1-score	support
B-FACILITY	1.00	0.70	0.82	10
B-GPE	0.91	0.54	0.68	289
B-GSP	0.00	0.00	0.00	6
B-LOCATION	0.00	0.00	0.00	7
B-ORGANIZATION	0.78	0.39	0.52	228
B-PERSON	0.77	0.49	0.60	299
I-FACILITY	1.00	0.58	0.74	12
I-GPE	0.75	0.82	0.79	40
I-GSP	0.00	0.00	0.00	1
I-LOCATION	0.00	0.00	0.00	8
I-ORGANIZATION	0.66	0.64	0.65	151
I-PERSON	0.54	0.71	0.61	173
O	0.97	0.99	0.98	14220
accuracy			0.96	15444
macro avg	0.57	0.45	0.49	15444
weighted avg	0.95	0.96	0.95	15444

accuracy: 0.9573297073297073

نتایج بالا نشان می دهد که اکثر تگ ها مربوط به O بوده اند و دقت بالای مدل نیز به همین خاطر است.کم بودن سایر تگ ها باعث شده تا به طور مجزا دقت بالایی بر روی سایر کلاس ها به دست نیاید گرچه با تعداد داده های

موجود برای برخی کلاس ها مانند B-Facility و یا I-GPE معیار ها مقدار مطلوبی دارند اما در صورتی که تنوع تگ ها بیشتر بود و یا داده های بیشتری در اختیار داشتیم احتمالا در سایر کلاس ها نیز به دقت بهتری می رسیدیم.

ت)

برای مسائل ner از شبکه های عصبی بازگشتی نیز می توان استفاده کرد. اما یک شبکه عصبی به این صورت کار می کند که پس از دریافت embedding کلمات در لایه آخر احتمالات مربوط به هر گروه اسمی را به دست می آورد و سپس با روش های مختلف مانند محاسبه بیشنه احتمال تگ را مشخص میکند. اما در مسائل ner لازم است تا عدم وجود توالی های خاصی چک شود به این منظور باید لایه ای دیگر اضافه کرد تا با گرفتن احتمالات و دریافت توالی های غیرمجاز تگ ها را در خروجی باز گرداند(روش مشابه در قسمت قبل که احتمالات به صفر تبدیل شد). استفاده از CRF نیز می تواند با افزودن یک لایه به شبکه این مشکل را برطرف کند.