بنام خدا

دانشکده‌ی مهندسی برق و کامپیوتر

درس پردازش زبان‌های طبیعی

شایان واصف احمدزاده

شماره دانشجویی : 810197603

تمرین کامپیوتری 3

روز آپلود : 28 فروردین

POS/NER

استاد : دکتر فِیلی

Table of Contents

تعیین نقش کلمات

الف)

تفاوت خواندن دادگان در حالت عادی نسبت به Universal در این است که tag های نسبت داده شده در حالت عادی، پر جزیئات تر از حالت Universal هستند. برای مثال در حالت Universal، برای تمامی لغات که از جنس اسم هستند، لیبل NOUN اختصاص داده می‌شود درحالیکه در حالت عادی، بسته به اینکه اسم از نوع مفرد یا جمع یا حالات دیگر باشد، لیبل های مجزایی در نظر گرفته می‌شود.

در شکل 1، Tag های مربوط به حالت عادی و در شکل 2، Tag های مربوط به حالت Universal خوانده شده است. همانطور که مشاهده می‌شود، تعداد Tag های مربوط به حالت عادی دقیق تر می‌باشد.

CC	Coordinating conjunction	NNS	Noun, plural	UH	Interjection
CD	Cardinal number	NNP	Proper noun, singular	VB	Verb, base form
DT	Determiner	NNPS	Proper noun, plural	VBD	Verb, past tense
EX	Existential there	PDT	Predeterminer	VBG	Verb, gerund or present participle
FW	Foreign word	POS	Possessive ending	VBN	Verb, past participle
IN	Preposition or subordinating conjunction	PRP	Personal pronoun	VBP	Verb, non-3rd person singular present
JJ	Adjective	PRP$	Possessive pronoun	VBZ	Verb, 3rd person singular present
JJR	Adjective, comparative	RB	Adverb	WDT	Wh-determiner
JJS	Adjective, superlative	RBR	Adverb, comparative	WP	Wh-pronoun
LS	List item marker	RBS	Adverb, superlative	WP$	Possessive wh-pronoun
MD	Modal	RP	Particle	WRB	Wh-adverb
NN	Noun, singular or mass	SYM	Symbol		
		TO	to		

شکل 1: Normal's Tag

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

شکل 2 : Universal's Tag

همچنین در شکل 3، برای یک جمله خاص، Tag های مربوط به حالت Universal و Normal آورده شده است :

index	Words	Normal	Universal
0	A	DT	DET
1	form	NN	NOUN
2	of	IN	ADP
3	asbestos	NN	NOUN
4	once	RB	ADV
5	used	VBN	VERB
6	*	-NONE-	X
7	*	-NONE-	X
8	to	TO	PRT
9	make	VB	VERB
10	Kent	NNP	NOUN
11	cigarette	NN	NOUN
12	filters	NNS	NOUN
13	has	VBZ	VERB
14	caused	VBN	VERB
15	a	DT	DET
16	high	JJ	ADJ
17	percentage	NN	NOUN
18	of	IN	ADP
19	cancer	NN	NOUN
20	deaths	NNS	NOUN
21	among	IN	ADP
22	a	DT	DET
23	group	NN	NOUN
24	of	IN	ADP

index	Words	Normal	Universal
25	workers	NNS	NOUN
26	exposed	VBN	VERB
27	*	-NONE-	X
28	to	TO	PRT
29	it	PRP	PRON
30	more	RBR	ADV
31	than	IN	ADP
32	30	CD	NUM
33	years	NNS	NOUN
34	ago	IN	ADP
35	.	.	.
36	researchers	NNS	NOUN
37	reported	VBD	VERB
38	0	-NONE-	X
39	*T*-1	-NONE-	X
40	.	.	.

شکل 3 : Universal vs Normal Tagging for a random sentence

ب)

در ابتدا ترتیب لیست جملات خوانده شده را بهم می‌ریزیم [1] و سپس 80 درصد اول داده ها را به آموزش[2]، 10 درصد بعدی را به داده صحت [3]، و 10 درصد پایانی را به دادگان آزمون[4] اختصاص می‌دهیم :

```
# get the corpus
import random
sentences=list(treebank.tagged_sents(tagset='universal'))
random.shuffle(sentences)
#Splitting the data for train and test
split_num_train = int(len(sentences)*0.8)
split_num_valid = int(len(sentences)*0.9)
train_data = sentences[0:split_num_train]
valid_data = sentences[split_num_train:split_num_valid]
test_data  = sentences[split_num_valid:]
```

[1] Shuffle
[2] Train Data
[3] Validation Data
[4] Test Data

پ)

شبه کد[5] الگوریتم ویتربی[6]، به صورت شکل 4 می‌باشد که من طبق آن در کد پیاده سازی کردم :

Algorithm 11 The Viterbi algorithm. Each $s_m(k, k')$ is a local score for tag $y_m = k$ and $y_{m-1} = k'$.

for $k \in \{0, \dots K\}$ **do**
  $v_1(k) = s_1(k, \Diamond)$
for $m \in \{2, \dots, M\}$ **do**
  for $k \in \{0, \dots, K\}$ **do**
    $v_m(k) = \max_{k'} s_m(k, k') + v_{m-1}(k')$
    $b_m(k) = \text{argmax}_{k'} s_m(k, k') + v_{m-1}(k')$
$y_M = \text{argmax}_k s_{M+1}(\blacklozenge, k) + v_M(k)$
for $m \in \{M-1, \dots 1\}$ **do**
  $y_m = b_m(y_{m+1})$
return $\boldsymbol{y}_{1:M}$

شکل 4 : شبه کد الگوریتم ویتربی

در ادامه مرحله به مرحله، به بررسی پیاده سازی الگوریتم می‌پردازیم :

در ابتدا نیاز است که لغات را از Tag ها جدا کنیم و همچنین تشکیل یک Vocabulary دهیم که در مقابل هر لغت، اندیس دلخواهی نسبت دهد. همچنین تعداد ظاهر شدن هر کلمه را به صورت یک متغیر[7] در نظر گرفتیم و تنها لغاتی که بیش از 3 بار تکرار شده‌اند را در Vocabulary اضافه کردیم. همچنین به انتهای Vocabulary، کلمه ای به عنوان "UNK" اضافه کردیم و اندیس آخر را به آن اختصاص دادیم تا بعدا از آن استفاده کنیم.

در مرحله بعد، ماتریس های Transition و Emission را شکل دادیم :

index	START	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB	X	END
START	0.0	262.0	132.0	410.0	159.0	167.0	726.0	917.0	28.0	223.0	3.0	29.0	75.0	0.0
.	0.0	616.0	268.0	447.0	320.0	375.0	817.0	1178.0	715.0	314.0	18.0	801.0	176.0	3107.0
ADJ	0.0	321.0	328.0	376.0	25.0	77.0	23.0	3496.0	102.0	3.0	54.0	56.0	102.0	0.0
ADP	0.0	306.0	785.0	127.0	101.0	6.0	2410.0	2447.0	476.0	521.0	10.0	63.0	264.0	2.0
ADV	0.0	281.0	341.0	287.0	199.0	17.0	173.0	69.0	84.0	30.0	36.0	854.0	59.0	1.0
CONJ	0.0	53.0	203.0	73.0	104.0	1.0	168.0	607.0	76.0	81.0	6.0	278.0	8.0	0.0
DET	0.0	124.0	1313.0	57.0	86.0	3.0	33.0	3959.0	133.0	23.0	1.0	208.0	333.0	1.0
NOUN	0.0	5503.0	271.0	4046.0	387.0	992.0	309.0	5533.0	211.0	104.0	989.0	3216.0	670.0	16.0
NUM	0.0	339.0	94.0	101.0	8.0	37.0	10.0	983.0	515.0	4.0	70.0	42.0	583.0	2.0
PRON	0.0	83.0	153.0	48.0	53.0	9.0	19.0	444.0	14.0	16.0	32.0	919.0	182.0	0.0
PRT	0.0	107.0	225.0	59.0	28.0	5.0	271.0	627.0	143.0	48.0	4.0	1034.0	35.0	0.0
VERB	0.0	370.0	690.0	987.0	900.0	60.0	1465.0	1209.0	246.0	389.0	340.0	1823.0	2369.0	0.0
X	0.0	871.0	83.0	765.0	122.0	55.0	280.0	322.0	14.0	288.0	976.0	1064.0	410.0	1.0
END	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

شکل 5 : ماتریس Transition (بر حسب تعداد)

در زیر ماتریس Emission تا 20 ستون اول(20 لغت اول Vocabulary) رسم شده است :

[5] pseudocode
[6] Viterbi Algorithm
[7] Hyper parameter

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
START	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
.	3954.0	0.0	3066.0	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	562.0	560.0	0.0
ADJ	0.0	5.0	0.0	0.0	1.0	2.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ADP	0.0	0.0	0.0	1857.0	1.0	1.0	1265.0	0.0	0.0	0.0	0.0	0.0	415.0	0.0	636.0	0.0	0.0	0.0	0.0	0.0
ADV	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
CONJ	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1237.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
DET	0.0	3242.0	0.0	0.0	0.0	1510.0	0.0	0.0	0.0	0.0	0.0	0.0	244.0	0.0	0.0	0.0	571.0	0.0	0.0	0.0
NOUN	0.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
NUM	0.0	1.0	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
PRON	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
PRT	0.0	0.0	0.0	0.0	1735.0	0.0	13.0	0.0	0.0	0.0	0.0	614.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VERB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	74.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	554.0
X	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	907.0	878.0	785.0	0.0	0.0	652.0	0.0	584.0	0.0	0.0	0.0	0.0
END	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

شکل 6 : ماتریس Emission (بر حسب تعداد)

در مرحله بعد، این دو ماتریس را به ماتریس های احتمالی تبدیل می‌کنیم. همچنین برای Smoothing از ضریب α برای Transition Matrix و ضریب β برای Emission Matrix استفاده کردیم. در ادامه به کمک این دو ضریب مدل را بر روی دادگان صحت تنظیم[8] می‌کنیم.

*به دلیل اینکه اندازه ماتریس Transition بسیار کوچک تر از Emission هست و بنابراین تعداد صفر های ماتریس Emission بسیار بیشتر می‌باشد، نیاز است که پارامتر β را به نسبت بزرگتر از α انتخاب کنیم.

در نهایت الگوریتم ویتربی را مشابه شبه کد بالا پیاده سازی می‌کنیم. در ماتریس v موجود در الگوریتم، بهترین احتمال بدست آمده در هر مرحله را ذخیره می‌کنیم و در ماتریس b، اندیس های متناظر با Tag ها را که بیشترین احتمال را در هر مرحله می‌دهند را ذخیره می‌کنیم تا در آخر بتوانیم بهترین مسیر را بازیابی کنیم.

حال به ازای $\alpha = [0.01 , 0.02 , 0.03] , \beta = [1 , 2 ,3]$، دقت مدل را بر روی دادگان صحت بررسی می‌کنیم تا بهترین زوج را انتخاب کنیم :

alpha=0.01, beta=1.0, overal_accuracy=0.9197757390417941
alpha=0.01, beta=2.0, overal_accuracy=0.9197757390417941
alpha=0.01, beta=3.0, overal_accuracy=0.9196738022426095
alpha=0.02, beta=1.0, overal_accuracy=0.9200815494393476
alpha=0.02, beta=2.0, overal_accuracy=0.9200815494393476
alpha=0.02, beta=3.0, overal_accuracy=0.9198776758409786
alpha=0.03, beta=1.0, overal_accuracy=0.9198776758409786
alpha=0.03, beta=2.0, overal_accuracy=0.9195718654434251
alpha=0.03, beta=3.0, overal_accuracy=0.919367991845056

طبق نتایج بالا، $\alpha = 0.02 , \beta = 2$، انتخاب مناسبی می‌باشد.

حال دقت را بر روی دادگان آزمون بدست می‌آوریم :

alpha=0.01, beta=2, overal_accuracy=0.9115158753856872

[8] Fine Tuning

ت)

در ادامه 5 جمله رندوم از Corpus را خوانده‌ایم، سپس Tag های واقعی و پیش بینی هر کدام را چاپ کردیم و سپس برای کلماتی که مقدار واقعی با پیشبینی شده مغایرت دارد، مقدار واقعی را رنگ قرمز و مقدار پیشبینی شده اشتباه را با رنگ آبی آورده‌ایم. در نهایت دقت مدل بر روی هر کدام از جملات را نیز چاپ کردیم :

```
###########
 The sentence:   Other paper and forest-products stocks closed *-1 mixed.

 Actual labels: [('Other', 'ADJ'), ('paper', 'NOUN'), ('and', 'CONJ'),
('forest-products', 'NOUN'), ('stocks', 'NOUN'), ('closed', 'VERB'), ('*-
1', 'X'), ('mixed', 'VERB'), ('.', '.')]

 Predicted labels: [('Other', 'NOUN'), ('paper', 'NOUN'), ('and', 'CONJ'),
('forest-products', 'ADJ'), ('stocks', 'NOUN'), ('closed', 'VERB'), ('*-
1', 'X'), ('mixed', 'VERB'), ('.', '.')]

 Actual tag: ('Other', 'ADJ'), predicted tag :('Other', 'NOUN')

 Actual tag: ('forest-products', 'NOUN'), predicted tag :('forest-
products', 'ADJ')


alpha=0.01, beta=2, overal_accuracy=0.7777777777777778

###########
 The sentence: PRECIOUS METALS: Futures prices eased as increased
stability and strength came into the securities markets.

 Actual labels: [('PRECIOUS', 'NOUN'), ('METALS', 'NOUN'), (':', '.'),
('Futures', 'NOUN'), ('prices', 'NOUN'), ('eased', 'VERB'), ('as', 'ADV'),
('increased', 'VERB'), ('stability', 'NOUN'), ('and', 'CONJ'),
('strength', 'NOUN'), ('came', 'VERB'), ('into', 'ADP'), ('the', 'DET'),
('securities', 'NOUN'), ('markets', 'NOUN'), ('.', '.')]

 Predicted labels: [('PRECIOUS', 'NOUN'), ('METALS', 'NOUN'), (':', '.'),
('Futures', 'NOUN'), ('prices', 'NOUN'), ('eased', 'VERB'), ('as', 'ADV'),
('increased', 'VERB'), ('stability', 'NOUN'), ('and', 'CONJ'),
('strength', 'NOUN'), ('came', 'VERB'), ('into', 'ADP'), ('the', 'DET'),
('securities', 'NOUN'), ('markets', 'NOUN'), ('.', '.')]


alpha=0.01, beta=2, overal_accuracy=1.0

###########
 The sentence: As San Francisco digs out from The Pretty Big One,
opponents say 0 the last thing 0 the city can afford *T*-1 is an expensive
new stadium.

 Actual labels : [('As', 'ADP'), ('San', 'NOUN'), ('Francisco', 'NOUN'),
('digs', 'VERB'), ('out', 'PRT'), ('from', 'ADP'), ('The', 'DET'),
('Pretty', 'NOUN'), ('Big', 'NOUN'), ('One', 'NUM'), (',', '.'),
('opponents', 'NOUN'), ('say', 'VERB'), ('0', 'X'), ('the', 'DET'),
```

```
('last', 'ADJ'), ('thing', 'NOUN'), ('0', 'X'), ('the', 'DET'), ('city',
'NOUN'), ('can', 'VERB'), ('afford', 'VERB'), ('*T*-1', 'X'), ('is',
'VERB'), ('an', 'DET'), ('expensive', 'ADJ'), ('new', 'ADJ'), ('stadium',
'NOUN'), ('.', '.')]

 Predicted labels : [('As', 'ADP'), ('San', 'NOUN'), ('Francisco',
'NOUN'), ('digs', 'NOUN'), ('out', 'PRT'), ('from', 'ADP'), ('The',
'DET'), ('Pretty', 'ADJ'), ('Big', 'NOUN'), ('One', 'NUM'), (',', '.'),
('opponents', 'NOUN'), ('say', 'VERB'), ('0', 'X'), ('the', 'DET'),
('last', 'ADJ'), ('thing', 'NOUN'), ('0', 'X'), ('the', 'DET'), ('city',
'NOUN'), ('can', 'VERB'), ('afford', 'VERB'), ('*T*-1', 'X'), ('is',
'VERB'), ('an', 'DET'), ('expensive', 'ADJ'), ('new', 'ADJ'), ('stadium',
'NOUN'), ('.', '.')]

 Actual tag: ('digs', 'VERB'), predicted tag :('digs', 'NOUN')

 Actual tag: ('Pretty', 'NOUN'), predicted tag :('Pretty', 'ADJ')


alpha=0.01, beta=2, overal_accuracy=0.9310344827586207

###########
 The sentence: In the classroom, students say 0 *T*-3, Mrs. Yeargin
distinguished herself by * varying teaching approaches -- *-1 forcing kids
to pair up *-2 to complete classroom work or using college-bowl type
competitions.

 Actual labels : [('In', 'ADP'), ('the', 'DET'), ('classroom', 'NOUN'),
(',', '.'), ('students', 'NOUN'), ('say', 'VERB'), ('0', 'X'), ('*T*-3',
'X'), (',', '.'), ('Mrs.', 'NOUN'), ('Yeargin', 'NOUN'), ('distinguished',
'VERB'), ('herself', 'PRON'), ('by', 'ADP'), ('*', 'X'), ('varying',
'VERB'), ('teaching', 'NOUN'), ('approaches', 'NOUN'), ('--', '.'), ('*-
1', 'X'), ('forcing', 'ADJ'), ('kids', 'NOUN'), ('to', 'PRT'), ('pair',
'VERB'), ('up', 'PRT'), ('*-2', 'X'), ('to', 'PRT'), ('complete', 'VERB'),
('classroom', 'NOUN'), ('work', 'NOUN'), ('or', 'CONJ'), ('using',
'VERB'), ('college-bowl', 'NOUN'), ('type', 'NOUN'), ('competitions',
'NOUN'), ('.', '.')]

 Predicted labels : [('In', 'ADP'), ('the', 'DET'), ('classroom', 'NOUN'),
(',', '.'), ('students', 'NOUN'), ('say', 'VERB'), ('0', 'X'), ('*T*-3',
'X'), (',', '.'), ('Mrs.', 'NOUN'), ('Yeargin', 'NOUN'), ('distinguished',
'NOUN'), ('herself', 'NOUN'), ('by', 'ADP'), ('*', 'X'), ('varying',
'VERB'), ('teaching', 'ADJ'), ('approaches', 'NOUN'), ('--', '.'), ('*-1',
'X'), ('forcing', 'VERB'), ('kids', 'NOUN'), ('to', 'PRT'), ('pair',
'VERB'), ('up', 'ADV'), ('*-2', 'X'), ('to', 'PRT'), ('complete', 'VERB'),
('classroom', 'ADJ'), ('work', 'NOUN'), ('or', 'CONJ'), ('using', 'VERB'),
('college-bowl', 'ADJ'), ('type', 'NOUN'), ('competitions', 'NOUN'), ('.',
'.')]

 Actual tag: ('distinguished', 'VERB'), predicted tag :('distinguished',
'NOUN')

 Actual tag: ('herself', 'PRON'), predicted tag :('herself', 'NOUN')
```

```
 Actual tag: ('teaching', 'NOUN'), predicted tag :('teaching', 'ADJ')

 Actual tag: ('forcing', 'ADJ'), predicted tag :('forcing', 'VERB')

 Actual tag: ('up', 'PRT'), predicted tag :('up', 'ADV')

 Actual tag: ('classroom', 'NOUN'), predicted tag :('classroom', 'ADJ')

 Actual tag: ('college-bowl', 'NOUN'), predicted tag :('college-bowl',
'ADJ')


alpha=0.01, beta=2, overal_accuracy=0.8055555555555556

###########
 The sentence: Since then, a team of about 15 MITI and U.S. Commerce
Department officials have crossed the globe *-1 gauging consumer prices.

 Actual labels: [('Since', 'ADP'), ('then', 'ADV'), (',', '.'), ('a',
'DET'), ('team', 'NOUN'), ('of', 'ADP'), ('about', 'ADV'), ('15', 'NUM'),
('MITI', 'NOUN'), ('and', 'CONJ'), ('U.S.', 'NOUN'), ('Commerce', 'NOUN'),
('Department', 'NOUN'), ('officials', 'NOUN'), ('have', 'VERB'),
('crossed', 'VERB'), ('the', 'DET'), ('globe', 'NOUN'), ('*-1', 'X'),
('gauging', 'VERB'), ('consumer', 'NOUN'), ('prices', 'NOUN'), ('.', '.')]

 Predicted labels: [('Since', 'ADP'), ('then', 'ADV'), (',', '.'), ('a',
'DET'), ('team', 'NOUN'), ('of', 'ADP'), ('about', 'ADP'), ('15', 'NUM'),
('MITI', 'NOUN'), ('and', 'CONJ'), ('U.S.', 'NOUN'), ('Commerce', 'NOUN'),
('Department', 'NOUN'), ('officials', 'NOUN'), ('have', 'VERB'),
('crossed', 'VERB'), ('the', 'DET'), ('globe', 'NOUN'), ('*-1', 'X'),
('gauging', 'VERB'), ('consumer', 'NOUN'), ('prices', 'NOUN'), ('.', '.')]

 Actual tag: ('about', 'ADV'), predicted tag: ('about', 'ADP')


alpha=0.01, beta=2, overal_accuracy=0.9565217391304348
```

*طبق نتایج بالا، مدل در تشخیص کلماتی که با ‘ing’، ‘ed’ و یا ‘s’ تمام می‌شوند، دچار مشکل شده است زیر کلمات متنوعی از دو دسته ”VERB“ و ”NOUN“ وجود دارند که پایانی مشابه عبارت های بالا دارند و مدل در تشخیص آنها دچار اشتباه شده است.

* مدل همچنین در تشخیص بعضی کلمات بین ”NOUN“ و ”ADJ“ دچار مشکل شده است. برای مثال کلمه ”Other“ را به عنوان ”ADJ“ تشخیص داده در حالیکه ”NOUN“ می‌باشد.

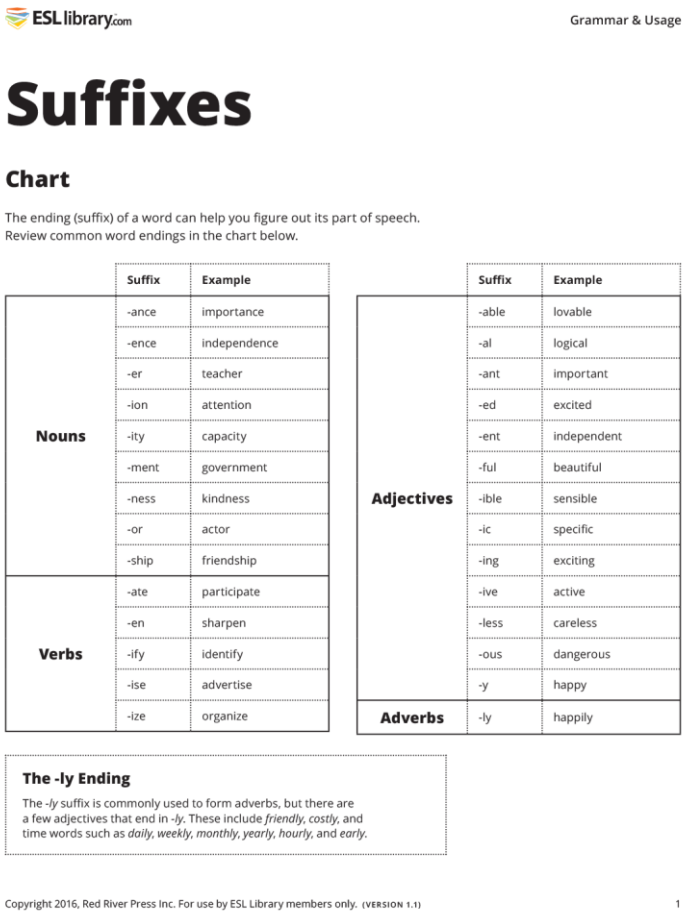
* همچنین در تشخیص کلماتی که به کمک “-” متشکل از دو کلمه جدا هستند (مانند 'forest-products') مدل به اشتباه افتاده است. دلیل این می‌تواند باشد که نوع Tag این کلمات می‌تواند کاملا از نوع Tag مربوط به هر دو کلمه تشکیل دهنده آن متفاوت باشد و در نتیجه از قانون خاصی تبعیت نمی‌کنند.

ث)

همانطور که در بخش پ بحث شد، در انتهای Vocabulary، کلمه "UKN" را به عنوان آخرین اندیس اضافه می‌کنیم . طبیعتا در ساخت ماتریس Emission از روی دادگان آموزش، تعداد تکرار مربوط به ستون "UKN" صفر خواهد بود ولی ستون آن ایجاد شده است. پس در برخورد با داده‌ی ناشناخته در دادگان صحت، احتمال کلمه "UKN" را محاسبه می‌کنیم. برای آنکه این احتمال در دادگان صحت برابر صفر نباشد از پارامتر β برای Smoothing در ماتریس Emission استفاده می‌کنیم.

*روش بهتر این خواهد بود که به کمک ریخت شناسی[9] در زبان انگلیسی، برای Tag های معروف مثل "VERB"، "NOUN" ، "ADJ" و "ADV"، تعدادی پسوند[10] تعریف کنیم که با دیدن کلمه‌ی ناشناخته، اگر پسوند یکی از چهار Tag بالا را داشت، آنرا به عنوان Tag شناخته شده در نظر بگیرد.

برای مثال، تعدادی از پسوندهای ممکن برای چهار Tag معروف بالا در شکل 7 آورده شده است :



ESL library.com

Grammar & Usage

Suffixes

Chart

The ending (suffix) of a word can help you figure out its part of speech.
Review common word endings in the chart below.

	Suffix	Example
Nouns	-ance	importance
	-ence	independence
	-er	teacher
	-ion	attention
	-ity	capacity
	-ment	government
	-ness	kindness
	-or	actor
	-ship	friendship
Verbs	-ate	participate
	-en	sharpen
	-ify	identify
	-ise	advertise
	-ize	organize

	Suffix	Example
Adjectives	-able	lovable
	-al	logical
	-ant	important
	-ed	excited
	-ent	independent
	-ful	beautiful
	-ible	sensible
	-ic	specific
	-ing	exciting
	-ive	active
	-less	careless
	-ous	dangerous
	-y	happy
Adverbs	-ly	happily

The -ly Ending

The *-ly* suffix is commonly used to form adverbs, but there are a few adjectives that end in *-ly*. These include *friendly, costly,* and time words such as *daily, weekly, monthly, yearly, hourly,* and *early.*

شکل 7 : Some common suffixes

[9] Morphology
[10] Suffix

ج)

در ابتدا کل دادگان را به دو دسته X و Y تقسیم می‌کنیم که در دسته X تنها لغات و در دسته Y، تنها Tag ها قرار بگیرد :

```
print ('sample X: ', X [0], '\n') # Tokens
print ('sample Y: ', Y [0], '\n') # Tags
```

sample X:

['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'board', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']

sample Y:

['NOUN', 'NOUN', '.', 'NUM', 'NOUN', 'ADJ', '.', 'VERB', 'VERB', 'DET', 'NOUN', 'ADP', 'DET', 'ADJ', 'NOUN', 'NOUN', 'NUM', '.']

اطلاعات کلی زیر را از داده اولیه استخراج می‌کنیم :

The total number of tagged sentences: 3914, The Vocabulary size: 11387, The total number of tags: 12

در وهله بعد نیاز است که برای هر لغت موجود در X و هر Tag موجود در Y، یک عدد صحیح نسبت دهیم[11].

```
encoded_X= []
for lst in X :
  encoded_X.append([vocab_words[item.lower()] for item in lst])
encoded_Y = []
for lst in Y :
  encoded_Y.append([vocab_tags[item.lower()] for item in lst])
```

بعد از اینکه داده‌ها ما از جنس عدد شدند، نیاز است که طول تمامی جملات را یکسان کنیم[12] تا بتوانیم آنرا به ورودی مدل بدهیم. طول ثابتی که باید برای تمامی جملات در نظر گرفت یک متغیر است. در زیر، بیشترین طول جمله بدست آمده در Corpus را محاسبه کردیم :

```
# check length of longest sentence
lengths = [len(seq) for seq in encoded_X]
print("Length of longest sentence: {}".format(max(lengths))) : Length of longest sentence: 271
```

با کمی تحلیل آماری، بیشترین طول جمله مجاز را عدد 100 انتخاب می‌کنیم .

*نحوه کار به این صورت خواهد بود که جملاتی که طولشان از 100 کمتر است با اضافه کردن عدد صفر در ابتدا طولشان را افزایش می‌دهیم و در صورتی که طول جمله‌ای بیشتر از 100 باشد، 100 لغت اول آن جمله را در نظر می‌گیریم و بقیه را دور می‌اندازیم. برای مثال طول جدید Y_pad شده را مشاهده می‌کنید :

```
Y_pad.shape : (3914, 100)
```

[11] Encoding
[12] Padding

بعد از اینکه پیش‌پردازش های اولیه انجام شد، نوبت با کار با کتابخانه Pytorch[13] می‌رسد.
در ابتدا کلاس دیتاست خود را تعریف می‌کنیم :

```
# define a dataset class
class TBDataset(Dataset):
  def __init__(self ,  transform = None):
    #data loading

    self.x = X_pad
    self.y = Y_pad # [0] => reshaped already to accepted format

    self.n_samples = X_pad.shape[0]

    self.transform = transform

  def __getitem__(self, index):
    #indexing
      sample = self.x[index] , self.y[index]

      if self.transform :
        sample = self.transform(sample)

      return sample

  def __len__(self):
    return self.n_samples
```

همانطور که مشاهده می‌کنید، متغیر Transform به صورت اختیاری تعریف شده تا تغییرات احتمالی لازم بر روی داده اعمال گردد. حال کلاس OneHot را تعریف می‌کنیم تا لیبل ما به صورت one hot در بیاید. (*دقت کنید که Y_pad ما یک ماتریس می‌باشد، پس با اعمال این تبدیل، به یک Tensor تبدیل می‌شود.)

```
class OneHot:
  def __call__(self, sample) :
      inputs , targets = sample
      import numpy as np
      targets = targets.astype('int').reshape(-1)
      one_hot_targets = np.eye(num_tags+1)[targets]
      return torch.from_numpy(inputs) , torch.from_numpy(one_hot_targets)
```

*حال تبدیل بالا را در قالب پارامتر transform به کلاس دیتاست خود می‌دهیم و تعریف دیتاست ما به اتمام می‌رسد :

```
dataset = TBDataset(transform = OneHot())
```

[13] https://pytorch.org/docs/stable/index.html

حال نیاز است که به کمک زیرکتابخانه Random_splite، دیتاست بدست آمده را به سه بخش آموزش، صحت و آزمون تقسیم کنیم :

```
from torch.utils.data import random_split
train_dataset, test_dataset = random_split(dataset, [int(dataset.n_samples*0.85), dataset.n_samples- int(dataset.n_samples*0.85)])
train_dataset, validation_dataset = random_split(train_dataset, [int(len(train_dataset)*0.85), len(train_dataset)-int(len(train_dataset)*0.85)])
```

حال که به دادگان آموزش دسترسی داریم، نیاز که قبل از آموزش مدل، آنرا به دسته‌های با طول مشخص تقسیم کنیم. به عنوان یک متغیر ثابت، دسته ها را 64 تایی در نظر می‌گیریم :

```
train_loader = DataLoader(train_dataset , batch_size = 64 ,shuffle = True)
valid_loader = DataLoader(validation_dataset , batch_size = 64 ,shuffle = True)
test_loader = DataLoader(test_dataset , batch_size = 64 ,shuffle = True)
```

برای مثال، اندازه داده ورودی و لیبل را برای یک دسته از دادگان آموزش بررسی می‌کنیم :

```
features , labels  = next(iter(train_loader))
print(features.shape , "\n" , labels.shape )
```

```
torch.Size([64, 100])
 torch.Size([64, 100, 13])
```

*دقت کنید که چون لیبل های ما نیز در مرحله قبل Pad شده‌اند ، با اضافه شدن عدد صفر، تعداد Tag ها برابر 13 می‌باشد.

در ابتدا برای آموزش مدل، پارامتر های زیر را تنظیم می‌کنیم :

```
# init
EMBEDDING_SIZE  = 300
VOCABULARY_SIZE = num_words
hidden_size = 512
num_layers = 1
learning_rate = 0.002
num_classes = num_tags+1
num_epochs = 5
```

همچنین قبل آموزش مدل، وجود "cuda" را چک می‌کنیم :

```
# device config
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

در نهایت کلاس RNN را ایجاد می‌کنیم و مدل را آموزش می‌دهیم :

```
class RNN(nn.Module):
    def __init__(self, vocab_size, embed_size, hidden_size, num_layers , num_classes):
        super(RNN, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size
        self.embed = nn.Embedding(vocab_size, embed_size)
```

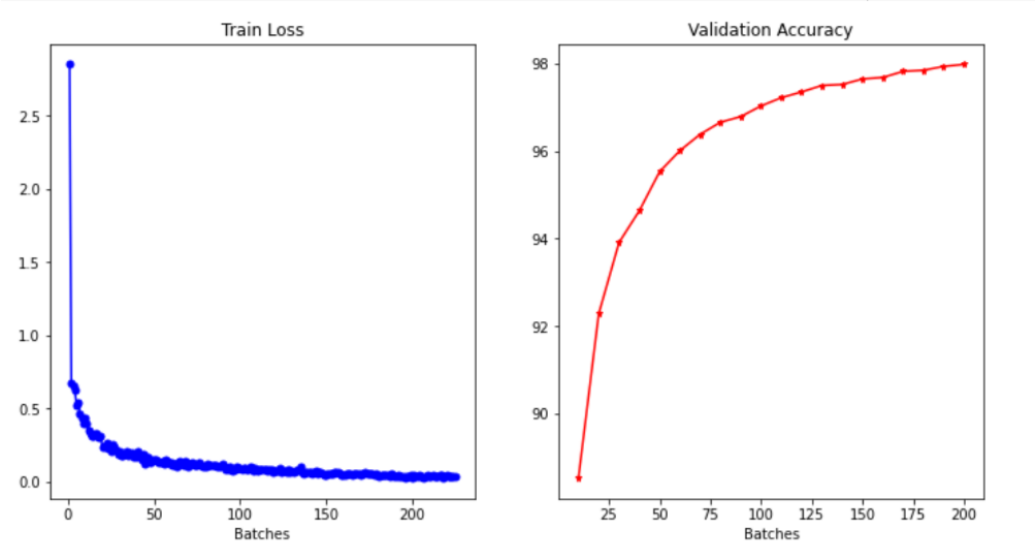
```
        self.rnn = nn.RNN(embed_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size,  num_classes)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)
        # Embed word ids to vectors
        x = self.embed(x)
        #print(x.shape)
        out, _ = self.rnn(x, h0)
        #print(out.shape)
        # Reshape output to (batch_size*sequence_length, hidden_size)
        #out = out[:, -1, :]
        #print(out.shape)
        # Decode hidden states of all time steps
        out = self.fc(out)
        #print(out.shape)
        return out

model = RNN(VOCABULARY_SIZE, EMBEDDING_SIZE, hidden_size, num_layers , num_classes).to(devi
ce)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

دقت مدل بر روی دادگان صحت، به صورت زیر می‌باشد :

```
Epoch [1/5], Step [10/45], Loss: 0.4365
Accuracy of the network on the valid dataset of length 499 : 88.52304609218437 %
Epoch [1/5], Step [20/45], Loss: 0.2376
Accuracy of the network on the valid dataset of length 499 : 92.27855711422846 %
Epoch [1/5], Step [30/45], Loss: 0.2105
Accuracy of the network on the valid dataset of length 499 : 93.90581162324649 %
Epoch [1/5], Step [40/45], Loss: 0.1721
Accuracy of the network on the valid dataset of length 499 : 94.63326653306613 %
Epoch [2/5], Step [10/45], Loss: 0.1412
Accuracy of the network on the valid dataset of length 499 : 95.5250501002004 %
Epoch [2/5], Step [20/45], Loss: 0.1455
Accuracy of the network on the valid dataset of length 499 : 96.00801603206413 %
Epoch [2/5], Step [30/45], Loss: 0.1165
Accuracy of the network on the valid dataset of length 499 : 96.38076152304609 %
Epoch [2/5], Step [40/45], Loss: 0.1110
Accuracy of the network on the valid dataset of length 499 : 96.65731462925852 %
Epoch [3/5], Step [10/45], Loss: 0.0894
Accuracy of the network on the valid dataset of length 499 : 96.78557114228457 %
Epoch [3/5], Step [20/45], Loss: 0.0746
Accuracy of the network on the valid dataset of length 499 : 97.03006012024048 %
Epoch [3/5], Step [30/45], Loss: 0.0740
```

Accuracy of the network on the valid dataset of length 499 : 97.21843687374749 %
Epoch [3/5], Step [40/45], Loss: 0.0731
Accuracy of the network on the valid dataset of length 499 : 97.35070140280561 %
Epoch [4/5], Step [10/45], Loss: 0.0590
Accuracy of the network on the valid dataset of length 499 : 97.49899799599199 %
Epoch [4/5], Step [20/45], Loss: 0.0674
Accuracy of the network on the valid dataset of length 499 : 97.52104208416834 %
Epoch [4/5], Step [30/45], Loss: 0.0494
Accuracy of the network on the valid dataset of length 499 : 97.64729458917836 %
Epoch [4/5], Step [40/45], Loss: 0.0566
Accuracy of the network on the valid dataset of length 499 : 97.6813627254509 %
Epoch [5/5], Step [10/45], Loss: 0.0380
Accuracy of the network on the valid dataset of length 499 : 97.81963927855712 %
Epoch [5/5], Step [20/45], Loss: 0.0305
Accuracy of the network on the valid dataset of length 499 : 97.84569138276554 %
Epoch [5/5], Step [30/45], Loss: 0.0330
Accuracy of the network on the valid dataset of length 499 : 97.93386773547094 %
Epoch [5/5], Step [40/45], Loss: 0.0436
Accuracy of the network on the valid dataset of length 499 : 97.97995991983969 %

همچنین نمودار دقت و خطا بر روی دادگان صحت، به صورت زیر می‌باشد :



شکل 8 : نمودار دقت و خطا بر روی دادگان صحت

سه متغیر زیر را به عنوان پارامتر های مدل در نظر می‌گیریم و بر روی دادگان صحت بررسی می‌کنیم :

[14]EMB = [50 ,100 ,150 ,300]
[15]NL = [1 , 2 , 3]
[16]HS = [64 ,128 ,512]

[14] Embedding Size
[15] Number of Layers
[16] Hidden Size

########## Embedding = 50 , Number of Layers = 1 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.1610

Accuracy of the network on the valid dataset of length 499 : 94.92785571142285 %

########## Embedding = 50 , Number of Layers = 1 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.1292
Accuracy of the network on the valid dataset of length 499 : 95.54909819639279 %

########## Embedding = 50 , Number of Layers = 1 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.1054
Accuracy of the network on the valid dataset of length 499 : 96.00400801603206 %

########## Embedding = 50 , Number of Layers = 2 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.1548
Accuracy of the network on the valid dataset of length 499 : 95.2244488977956 %

########## Embedding = 50 , Number of Layers = 2 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.1540
Accuracy of the network on the valid dataset of length 499 : 95.80761523046093 %

########## Embedding = 50 , Number of Layers = 2 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.1128
Accuracy of the network on the valid dataset of length 499 : 95.94789579158316 %

########## Embedding = 50 , Number of Layers = 3 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.1548

Accuracy of the network on the valid dataset of length 499 : 95.10420841683367 %

########## Embedding = 50 , Number of Layers = 3 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.1294
Accuracy of the network on the valid dataset of length 499 : 95.79759519038076 %

########## Embedding = 50 , Number of Layers = 3 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.1865
Accuracy of the network on the valid dataset of length 499 : 94.29859719438878 %

########## Embedding = 100 , Number of Layers = 1 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.1379
Accuracy of the network on the valid dataset of length 499 : 96.36472945891784 %

########## Embedding = 100 , Number of Layers = 1 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.0946
Accuracy of the network on the valid dataset of length 499 : 96.70941883767534 %

########## Embedding = 100 , Number of Layers = 1 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.0679
Accuracy of the network on the valid dataset of length 499 : 97.07414829659318 %

########## Embedding = 100 , Number of Layers = 2 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.1034
Accuracy of the network on the valid dataset of length 499 : 96.42284569138276 %

########## Embedding = 100 , Number of Layers = 2 , Hidden_size = 128##########

Epoch [5/5], Step [40/45], Loss: 0.0994
Accuracy of the network on the valid dataset of length 499 : 97.04208416833667 %

########## Embedding = 100 , Number of Layers = 2 , Hidden_size = 512##########

Epoch [5/5], Step [40/45], Loss: 0.0788
Accuracy of the network on the valid dataset of length 499 : 97.02004008016031 %

########## Embedding = 100 , Number of Layers = 3 , Hidden_size = 64##########

Epoch [5/5], Step [40/45], Loss: 0.1188
Accuracy of the network on the valid dataset of length 499 : 96.18036072144288 %

########## Embedding = 100 , Number of Layers = 3 , Hidden_size = 128##########

Epoch [5/5], Step [40/45], Loss: 0.0887
Accuracy of the network on the valid dataset of length 499 : 96.86573146292585 %

########## Embedding = 100 , Number of Layers = 3 , Hidden_size = 512##########

Epoch [5/5], Step [40/45], Loss: 0.0908
Accuracy of the network on the valid dataset of length 499 : 96.17835671342685 %

########## Embedding = 150 , Number of Layers = 1 , Hidden_size = 64##########

Epoch [5/5], Step [40/45], Loss: 0.0944
Accuracy of the network on the valid dataset of length 499 : 96.92384769539078 %

########## Embedding = 150 , Number of Layers = 1 , Hidden_size = 128##########

Epoch [5/5], Step [40/45], Loss: 0.0771
Accuracy of the network on the valid dataset of length 499 : 97.3567134268537 %

########## Embedding = 150 , Number of Layers = 1 , Hidden_size = 512##########

Epoch [5/5], Step [40/45], Loss: 0.0537
Accuracy of the network on the valid dataset of length 499 : 97.51503006012024 %

########## Embedding = 150 , Number of Layers = 2 , Hidden_size = 64##########

Epoch [5/5], Step [40/45], Loss: 0.0850
Accuracy of the network on the valid dataset of length 499 : 96.98196392785572 %

########## Embedding = 150 , Number of Layers = 2 , Hidden_size = 128##########

Epoch [5/5], Step [40/45], Loss: 0.0783
Accuracy of the network on the valid dataset of length 499 : 97.42284569138276 %

########## Embedding = 150 , Number of Layers = 2 , Hidden_size = 512##########

Epoch [5/5], Step [40/45], Loss: 0.0641
Accuracy of the network on the valid dataset of length 499 : 97.55711422845691 %

########## Embedding = 150 , Number of Layers = 3 , Hidden_size = 64##########

Epoch [5/5], Step [40/45], Loss: 0.0852
Accuracy of the network on the valid dataset of length 499 : 96.89779559118236 %

########## Embedding = 150 , Number of Layers = 3 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.0770
Accuracy of the network on the valid dataset of length 499 : 97.34468937875752 %

########## Embedding = 150 , Number of Layers = 3 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.0735
Accuracy of the network on the valid dataset of length 499 : 97.22044088176352 %

########## Embedding = 300 , Number of Layers = 1 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.0745
Accuracy of the network on the valid dataset of length 499 : 97.53106212424849 %

########## Embedding = 300 , Number of Layers = 1 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.0452
Accuracy of the network on the valid dataset of length 499 : 97.87575150300601 %

########## Embedding = 300 , Number of Layers = 1 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.0385
Accuracy of the network on the valid dataset of length 499 : 97.97194388777555 %

########## Embedding = 300 , Number of Layers = 2 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.0663
Accuracy of the network on the valid dataset of length 499 : 97.75751503006012 %

########## Embedding = 300 , Number of Layers = 2 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.0444
Accuracy of the network on the valid dataset of length 499 : 97.89579158316633 %

########## Embedding = 300 , Number of Layers = 2 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.0287
Accuracy of the network on the valid dataset of length 499 : 97.97595190380761 %

########## Embedding = 300 , Number of Layers = 3 , Hidden_size = 64#############

Epoch [5/5], Step [40/45], Loss: 0.0748
Accuracy of the network on the valid dataset of length 499 : 97.68336673346694 %

########## Embedding = 300 , Number of Layers = 3 , Hidden_size = 128#############

Epoch [5/5], Step [40/45], Loss: 0.0455
Accuracy of the network on the valid dataset of length 499 : 97.93186372745491 %

########## Embedding = 300 , Number of Layers = 3 , Hidden_size = 512#############

Epoch [5/5], Step [40/45], Loss: 0.0566
Accuracy of the network on the valid dataset of length 499 : 97.53907815631263 %

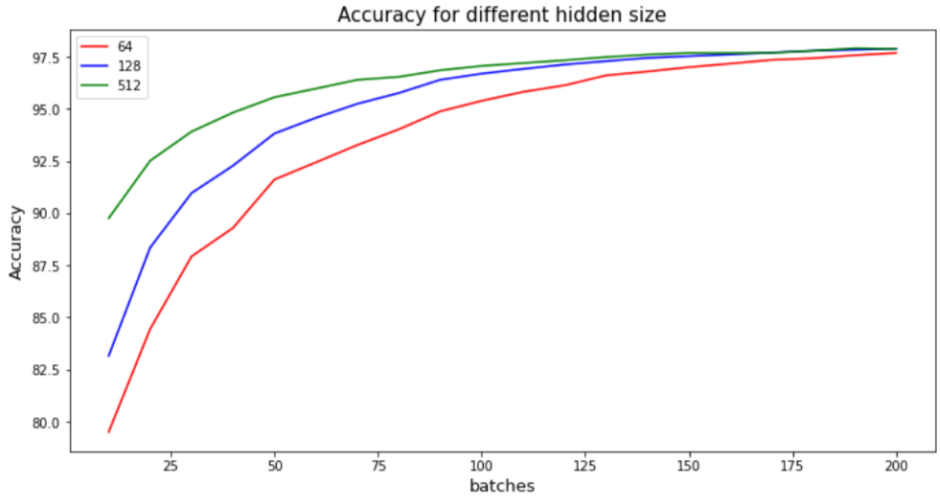
طبق خروجی‌های بدست آمده، بهترین پارامترها را انتخاب می‌کنیم:

```
np.where(Acc_valid_hyp == np.amax(Acc_valid_hyp))
 (array([3]), array([1]), array([2]))
```

Best params => embedding = 300 , num_layer = 2 ، Hidden_size = 512

حال اینبار تنها تاثیر اندازه لایه مخفی [17] را دردقت مدل می‌بینیم و رسم می‌کنیم :

HS = [64 ,128 ,512]



شکل 9 : نمودار دقت برای اندازه‌های مختلف لایه مخفی

ح)

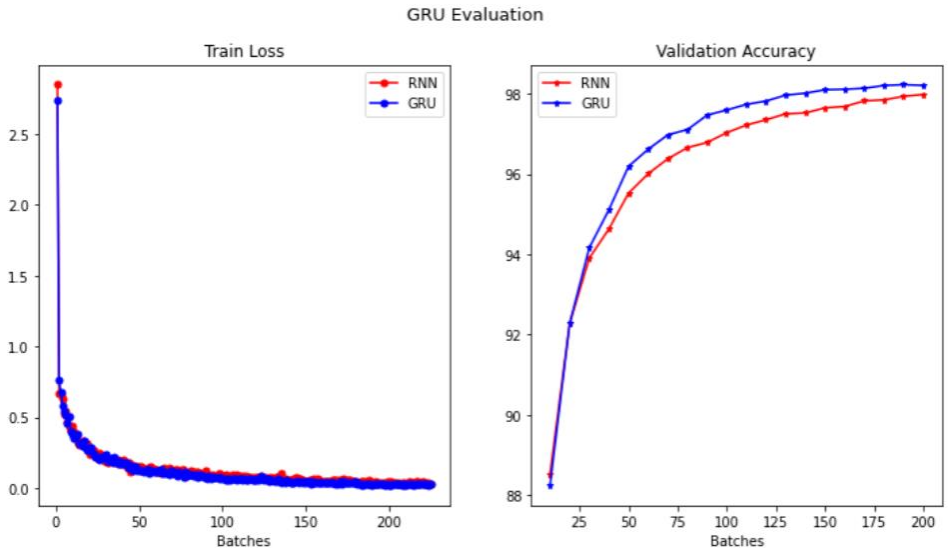
1. GRU

مشابه قسمت قبل اینبار کلاس GRU را تشکیل می‌دهیم و دقت بر روی دادگان صحت را با متغیرهای تنظیم شده در بخش قبل تکرار می‌کنیم :

Epoch [1/5], Step [10/45], Loss: 0.3846
Accuracy of the network on the valid dataset of length 499 : 88.24849699398797 %
Epoch [1/5], Step [20/45], Loss: 0.2828
Accuracy of the network on the valid dataset of length 499 : 92.27655310621242 %
Epoch [1/5], Step [30/45], Loss: 0.2344
Accuracy of the network on the valid dataset of length 499 : 94.16432865731463 %
Epoch [1/5], Step [40/45], Loss: 0.1770
Accuracy of the network on the valid dataset of length 499 : 95.11222444889779 %
Epoch [2/5], Step [10/45], Loss: 0.1322
Accuracy of the network on the valid dataset of length 499 : 96.19238476953907 %
Epoch [2/5], Step [20/45], Loss: 0.1142
Accuracy of the network on the valid dataset of length 499 : 96.61723446893788 %
Epoch [2/5], Step [30/45], Loss: 0.1086
Accuracy of the network on the valid dataset of length 499 : 96.97595190380761 %
Epoch [2/5], Step [40/45], Loss: 0.0767
Accuracy of the network on the valid dataset of length 499 : 97.1062124248497 %
Epoch [3/5], Step [10/45], Loss: 0.0672
Accuracy of the network on the valid dataset of length 499 : 97.46693386773548 %
Epoch [3/5], Step [20/45], Loss: 0.0664
Accuracy of the network on the valid dataset of length 499 : 97.59318637274549 %
Epoch [3/5], Step [30/45], Loss: 0.0600
Accuracy of the network on the valid dataset of length 499 : 97.73346693386773 %
Epoch [3/5], Step [40/45], Loss: 0.0608

[17] Hidden size

Accuracy of the network on the valid dataset of length 499 : 97.81763527054109 %
Epoch [4/5], Step [10/45], Loss: 0.0481
Accuracy of the network on the valid dataset of length 499 : 97.96392785571142 %
Epoch [4/5], Step [20/45], Loss: 0.0388
Accuracy of the network on the valid dataset of length 499 : 98.0120240480962 %
Epoch [4/5], Step [30/45], Loss: 0.0361
Accuracy of the network on the valid dataset of length 499 : 98.09819639278557 %
Epoch [4/5], Step [40/45], Loss: 0.0396
Accuracy of the network on the valid dataset of length 499 : 98.11022044088176 %
Epoch [5/5], Step [10/45], Loss: 0.0222
Accuracy of the network on the valid dataset of length 499 : 98.13426853707415 %
Epoch [5/5], Step [20/45], Loss: 0.0237
Accuracy of the network on the valid dataset of length 499 : 98.20440881763527 %
Epoch [5/5], Step [30/45], Loss: 0.0215
Accuracy of the network on the valid dataset of length 499 : 98.2244488977956 %
Epoch [5/5], Step [40/45], Loss: 0.0260
Accuracy of the network on the valid dataset of length 499 : 98.2064128256513 %



شکل 10 : نمودار دقت و خطا برای دو مدل RNN , GRU

و دوباره برای سه پارامتر زیر، مدل را تنظیم می‌کنیم :

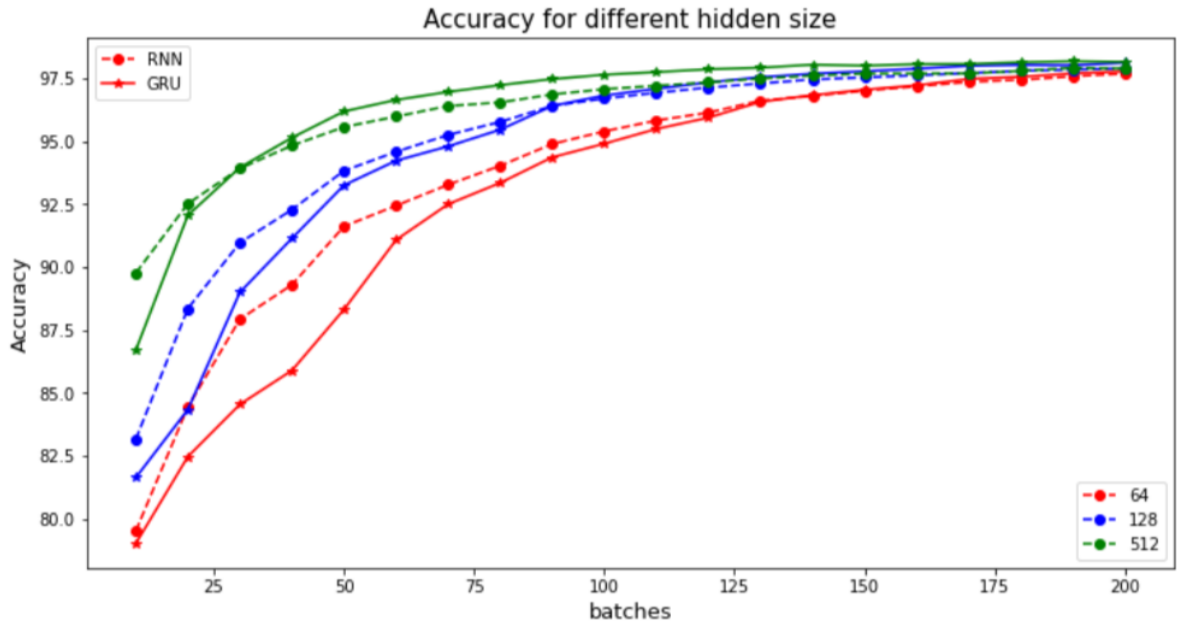
EMB = [50 ,100 ,150 ,300]
NL = [1 , 2 , 3]
HS = [64 ,128 ,512]

و بهترین سه تایی را بدست می‌آوریم :

np.where(Acc_valid_hyp == np.amax(Acc_valid_hyp))
(array([3]), array([0]), array([2]))
Best params => embedding = 300, num_layer = 1, Hidden_size = 512

*با پیچیده کردن مدل از RNN به GRU ، تعداد لایه‌های مخفی یکی کاهش یافت .

و دوباره تنها تاثیر اندازه لایه مخفی را دردقت مدل می‌بینیم و رسم می‌کنیم :



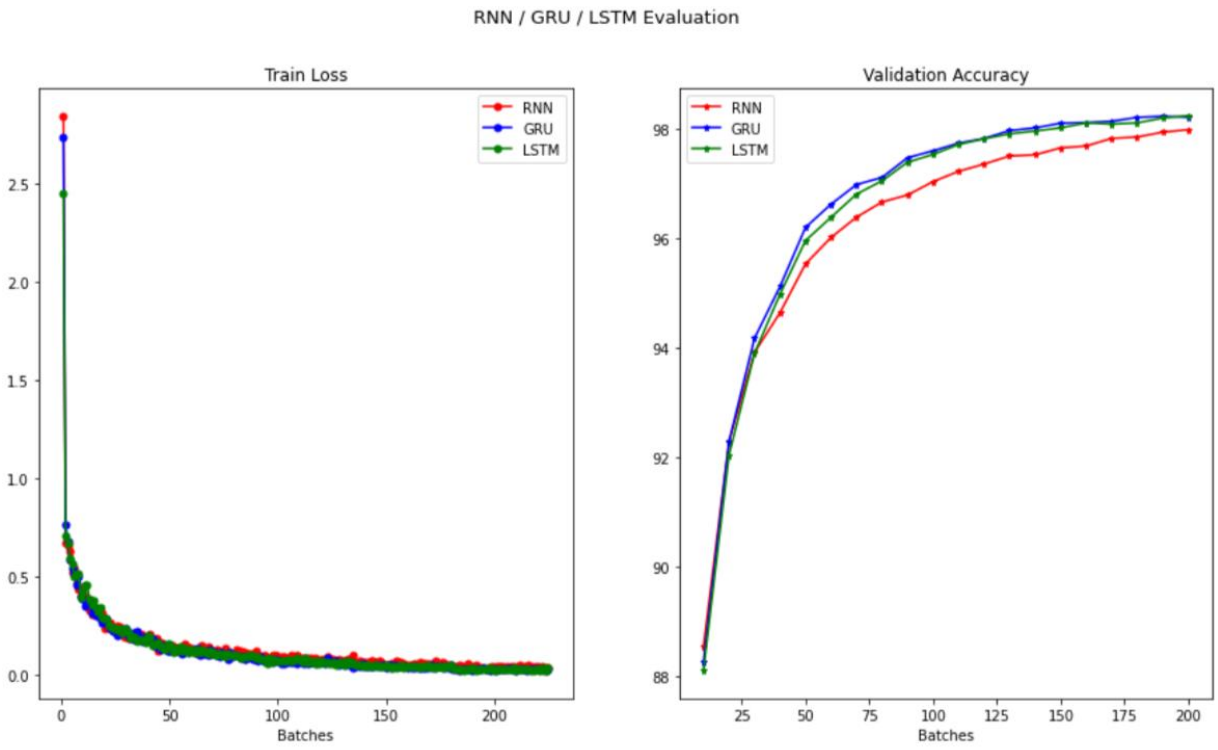
شکل 11 : نمودار دقت برای اندازه‌های مختلف لایه مخفی برای دو مدل GRU , RNN

2. LSTM

مشابه قسمت قبل اینبار کلاس LSTM را تشکیل می‌دهیم و دقت بر روی دادگان صحت را با متغیرهای تنظیم شده در بخش قبل تکرار می‌کنیم. تنها تفاوت در تعریف کلاس LSTM، تعریف cell state می‌باشد :

Epoch [1/5], Step [10/45], Loss: 0.4213
Accuracy of the network on the valid dataset of length 499 : 88.09218436873748 %
Epoch [1/5], Step [20/45], Loss: 0.2811
Accuracy of the network on the valid dataset of length 499 : 92.02605210420842 %
Epoch [1/5], Step [30/45], Loss: 0.2331
Accuracy of the network on the valid dataset of length 499 : 93.88977955911824 %
Epoch [1/5], Step [40/45], Loss: 0.1949
Accuracy of the network on the valid dataset of length 499 : 94.97194388777555 %
Epoch [2/5], Step [10/45], Loss: 0.1401
Accuracy of the network on the valid dataset of length 499 : 95.95190380761522 %
Epoch [2/5], Step [20/45], Loss: 0.1288
Accuracy of the network on the valid dataset of length 499 : 96.37675350701403 %
Epoch [2/5], Step [30/45], Loss: 0.1063
Accuracy of the network on the valid dataset of length 499 : 96.80160320641282 %
Epoch [2/5], Step [40/45], Loss: 0.0935
Accuracy of the network on the valid dataset of length 499 : 97.0440881763527 %
Epoch [3/5], Step [10/45], Loss: 0.0708
Accuracy of the network on the valid dataset of length 499 : 97.38476953907815 %
Epoch [3/5], Step [20/45], Loss: 0.0735

Accuracy of the network on the valid dataset of length 499 : 97.52705410821643 %
Epoch [3/5], Step [30/45], Loss: 0.0681
Accuracy of the network on the valid dataset of length 499 : 97.70941883767534 %
Epoch [3/5], Step [40/45], Loss: 0.0604
Accuracy of the network on the valid dataset of length 499 : 97.81362725450902 %
Epoch [4/5], Step [10/45], Loss: 0.0485
Accuracy of the network on the valid dataset of length 499 : 97.90380761523046 %
Epoch [4/5], Step [20/45], Loss: 0.0366
Accuracy of the network on the valid dataset of length 499 : 97.9559118236473 %
Epoch [4/5], Step [30/45], Loss: 0.0387
Accuracy of the network on the valid dataset of length 499 : 98.0120240480962 %
Epoch [4/5], Step [40/45], Loss: 0.0417
Accuracy of the network on the valid dataset of length 499 : 98.10420841683367 %
Epoch [5/5], Step [10/45], Loss: 0.0325
Accuracy of the network on the valid dataset of length 499 : 98.08416833667334 %
Epoch [5/5], Step [20/45], Loss: 0.0226
Accuracy of the network on the valid dataset of length 499 : 98.10420841683367 %
Epoch [5/5], Step [30/45], Loss: 0.0228
Accuracy of the network on the valid dataset of length 499 : 98.19238476953907 %
Epoch [5/5], Step [40/45], Loss: 0.0273
Accuracy of the network on the valid dataset of length 499 : 98.23446893787575 %



شکل 12 : نمودار دقت و خطا برای دو مدل RNN , GRU,LSTM

و دوباره برای سه پارامتر زیر، مدل را تنظیم می‌کنیم :

```
EMB = [50 ,100 ,150 ,300]
NL = [1 , 2 , 3]
HS = [64 ,128 ,512]
```

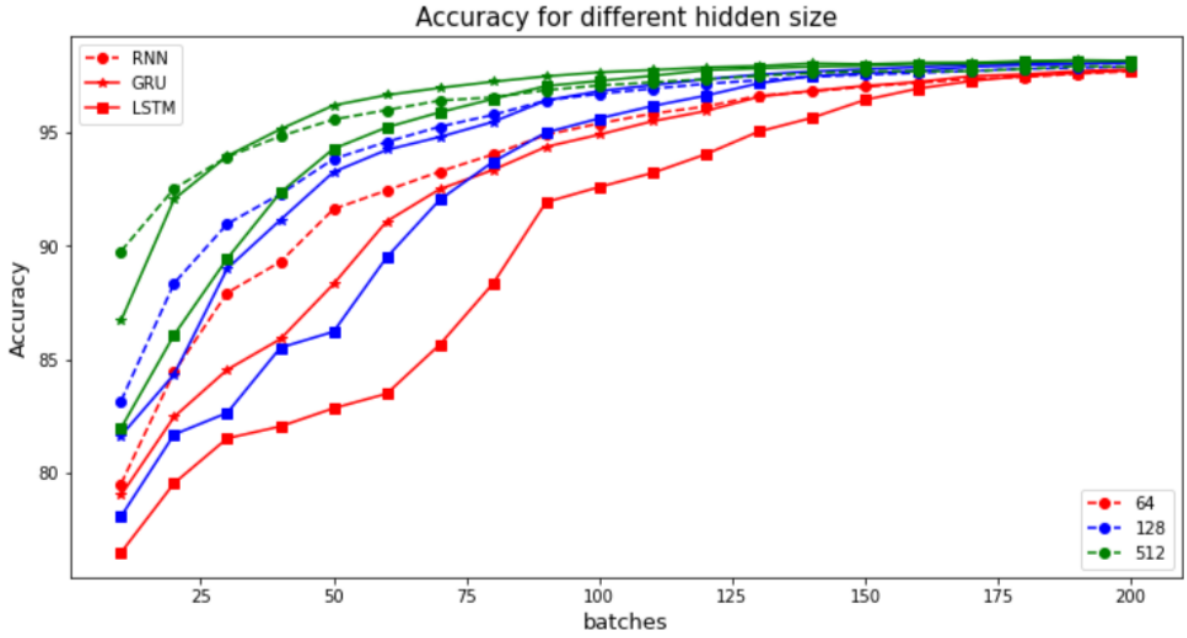
و بهترین سه تایی را بدست می‌آوریم :

```
np.where(Acc_valid_hyp == np.amax(Acc_valid_hyp))
(array([3]), array([0]), array([2]))
Best params => embedding = 300, num_layer = 1, Hidden_size = 512
```

*بهترین پارامتر های بدست آمده برای GRU و LSTM یکسان می‌باشد.

و دوباره تنها تاثیر اندازه لایه مخفی را دردقت مدل می‌بینیم و رسم می‌کنیم :

```
HS = [64 ,128 ,512]
```



شکل 13 : نمودار دقت برای اندازه‌های مختلف لایه مخفی برای دو مدل GRU , RNN, LSTM

*همانطور که مشاهده می‌کنید، برای لایه‌های 64 و 128، سرعت یادگیری RNN در 100 دسته ابتدایی بیشتر می‌باشد.

*از آنجایی که مدل‌های GRU و LSTM به مراتب پیچیدگی بیشتری نسبت به RNN دارند، پس برای تعداد لایه‌های 64 یا 128، برتری بیشتری ندارند. (*با پیچیده کردن مدل باید تعداد پارامترهای مدل هم افزایش داد)

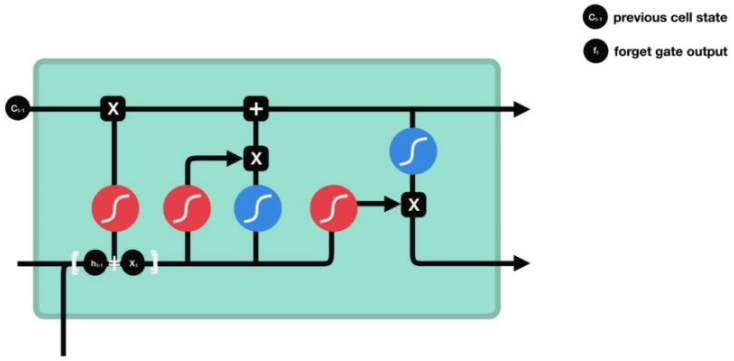
* با مدل های GRU و LSTM توانستیم به دقت بالای 98 دست پیدا کنیم.

خ)

هسته اصلی LSTM، cell state می‌باشد. برای آشنایی با کارایی cell state، نیاز است که با دو gate آشنا شویم :

1. Forget Gate:

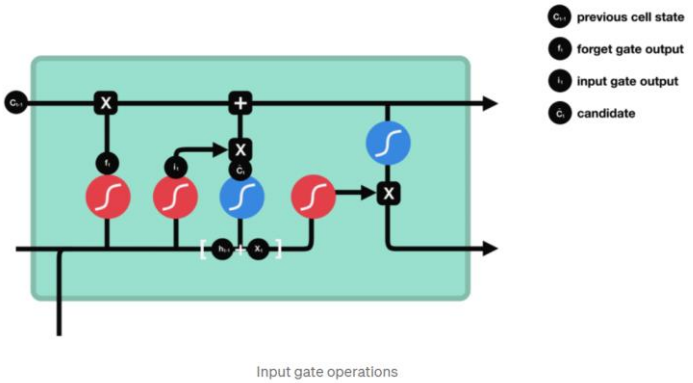
در این gate، تصمیم گرفته می‌شود که چه مقدار از اطلاعات گذشته به فراموشی سپرده شود. نحوه مار به این صورت است که ورودی جدید و لایه مخفی مرحله قبل با یکدیگر ادغام می‌شوند و از تابع sigmoid عبور می‌کنند و در cell state مرحله قبل ضرب می‌شود. خروجی تابع sigmoid بین دو عدد 0 و 1 می‌باشد. بنابراین اگه خروجی نزدیک به صفر باشد، تصمیم گرفته می‌شود که اطلاعات قبلی از cell state پاک شود و اگر نزدیک یه 1 باشد، تصمیم گرفته می‌شود که اکثر اطلاعات حفظ شود که در شکل 14 آورده شده است :



شکل 14 : Forget Gate

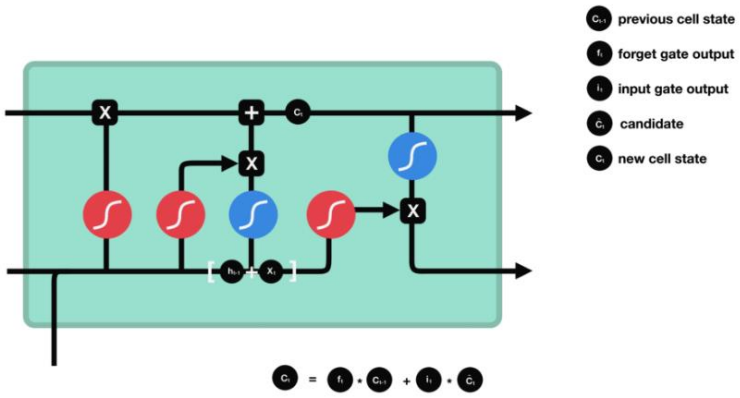
2. Input Gate:

در این gate تصمیم گرفته می‌شود که چه اطلاعات جدیدی نگه داشته شود و به cell state قبلی اضافه شود. نحوه کار به این شکل است که اطلاعات جدید (ادغام ورودی جدید و لایه مخفی قبلی) وارد تابع tah می‌شود تا بین بازه -1 تا 1 قرار گیرند. سپس باید توسط gate دیگری میزان بروز رسانی cell gate را کنترل کنیم. این کار توسط تایع sigmoid انجام می‌شود که که خروجی tanh را در عددی بین صفر و 1 ضرب می‌کند که در شکل 15 آورده شده است :



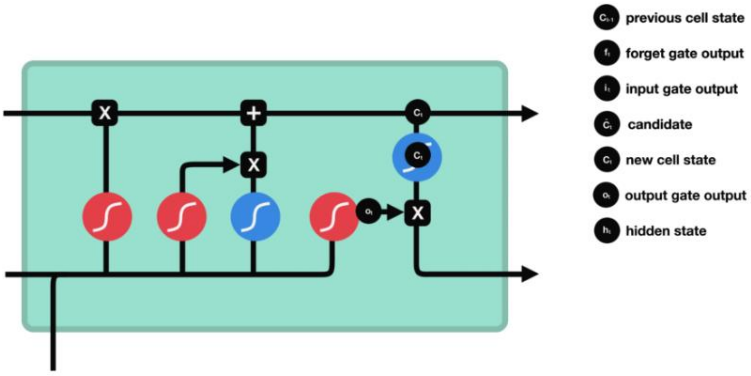
شکل 15 : Input Gate

بنابراین با آشنایی با دو gate بالا، حالا نحوه کار cell state بدست می‌آید. در واقع با جمع خروجی دو گیت بالا، cell state بروزرسانی می‌شود که در شکل 16 نشان داده شده است :



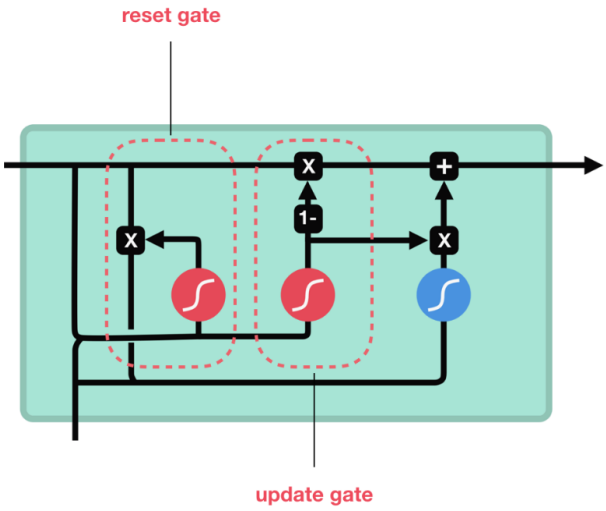
شکل 16 : cell state

گیت نهایی، output gate می‌باشد که در واقع اطلاعات لایه مخفی قبلی را برای لایه بعد برزورسانی می‌کند. برای این منظور اطلاعات بروز رسانی شده cell state در خروجی تابع sigmoid ضرب می‌شود تا میزان اطلاعات ذخیره شده برای لایه مخفی بعدی را بدست آورد.



شکل 17 : Output Gate

حال به بررسی GRU می‌پردازیم. GRU در واقع مدل ساده‌شده LSTM است که در بسیاری از کاربردها نتایج مشابه LSTM می‌دهد. GRU در واقع به دو gate کلی reset gate و update گیت تقسیم می‌شود که update گیت کارایی مشابه با forget gate و input fate در LSTM دارد و reset gate تصمیم می‌گیرد که مقدار از اطلاعات قبلی فراموش شود. در واقع در GRU از cell state استفاده نمی‌شود و اطلاعات توسط لایه مخفی بروزرسانی می‌شود. در شکل 18، نمایی کلی از GRU آورده شده است :



شکل 18 : GRU Architecture

د)

بهترین دقت بدست آمده از الگوریتم ویتربی برابر 0.911 و بهترین دقت بدست آمده از الگوریتم LSTM برابر 0.982 می‌باشد که نشان می‌دهد الگوریتم ویتربی چون از فرض مارکوف تنها استفاده می‌کند و وابستگی به حالت قبلی را تنها در نظر می‌گیرد، در بسیاری از حالت دچار مشکل می‌شود ولی مدل LSTM چون قادر به یادآوری و فراموشی اطلاعات است، گویی کل جمله را با هم در نظر می‌گیرد و طبعیتا دقت بسیار بهتری نیز می‌دهد.

تشخیص گروه‌های اسمی

الف)

```
# get the corpus
sentences=list(treebank.tagged_sents())
iob = [tree2conlltags(nltk.ne_chunk(sent)) for sent in  sentences]
iob[:2]
```

در ابتدا به کمک زیرکتابخانه ne_chunk ، برای هر جمله خروجی زیر را تولید می‌کنیم :

```
[[('Pierre', 'NNP', 'B-PERSON'),
  ('Vinken', 'NNP', 'B-ORGANIZATION'),
  (',', ',', 'O'),
  ('61', 'CD', 'O'),
  ('years', 'NNS', 'O'),
  ('old', 'JJ', 'O'),
  (',', ',', 'O'),
  ('will', 'MD', 'O'),
  ('join', 'VB', 'O'),
  ('the', 'DT', 'O'),
  ('board', 'NN', 'O'),
```

```
 ('as', 'IN', 'O'),
 ('a', 'DT', 'O'),
 ('nonexecutive', 'JJ', 'O'),
 ('director', 'NN', 'O'),
 ('Nov.', 'NNP', 'O'),
 ('29', 'CD', 'O'),
 ('.', '.', 'O')],
[('Mr.', 'NNP', 'B-PERSON'),
 ('Vinken', 'NNP', 'B-PERSON'),
 ('is', 'VBZ', 'O'),
 ('chairman', 'NN', 'O'),
 ('of', 'IN', 'O'),
 ('Elsevier', 'NNP', 'B-ORGANIZATION'),
 ('N.V.', 'NNP', 'O'),
 (',', ',', 'O'),
 ('the', 'DT', 'O'),
 ('Dutch', 'NNP', 'B-GPE'),
 ('publishing', 'VBG', 'O'),
 ('group', 'NN', 'O'),
 ('.', '.', 'O')]]
```

که از بین عنصر دوم و سوم tuple مربوط به هر لغت، تنها عنصر سوم را نیاز داریم :

```
sentences_ner = [[(item[0],item[2]) for item in sent] for sent in iob]
sentences_ner[:2]
```

```
[[('Pierre', 'B-PERSON'),
 ('Vinken', 'B-ORGANIZATION'),
 (',', 'O'),
 ('61', 'O'),
 ('years', 'O'),
 ('old', 'O'),
 (',', 'O'),
 ('will', 'O'),
 ('join', 'O'),
 ('the', 'O'),
 ('board', 'O'),
 ('as', 'O'),
 ('a', 'O'),
 ('nonexecutive', 'O'),
 ('director', 'O'),
 ('Nov.', 'O'),
 ('29', 'O'),
 ('.', 'O')],
[('Mr.', 'B-PERSON'),
 ('Vinken', 'B-PERSON'),
 ('is', 'O'),
 ('chairman', 'O'),
 ('of', 'O'),
 ('Elsevier', 'B-ORGANIZATION'),
 ('N.V.', 'O'),
 (',', 'O'),
 ('the', 'O'),
```

('Dutch', 'B-GPE'),
('publishing', 'O'),
('group', 'O'),
('.', 'O')]]

ب)

همانند سوال 1، ماتریس های Transition و Emission را بدست می‌آوریم :

	START	B-FACILITY	B-GPE	B-GSP	B-LOCATION	B-ORGANIZATION	B-PERSON	O	I-FACILITY	I-GPE	I-GSP	I-LOCATION	I-ORGANIZATION	I-PERSON	END
START	0.0	1.0	290.0	4.0	0.0	72.0	319.0	2445.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B-FACILITY	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	40.0	0.0	0.0	0.0	0.0	0.0	0.0
B-GPE	0.0	0.0	4.0	0.0	1.0	6.0	4.0	1028.0	0.0	197.0	0.0	0.0	0.0	0.0	3.0
B-GSP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	25.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0
B-LOCATION	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	22.0	0.0	0.0	0.0
B-ORGANIZATION	0.0	0.0	0.0	0.0	0.0	6.0	1.0	645.0	0.0	0.0	0.0	0.0	519.0	0.0	1.0
B-PERSON	0.0	0.0	0.0	0.0	0.0	2.0	0.0	479.0	0.0	0.0	0.0	0.0	0.0	834.0	2.0
O	0.0	33.0	1146.0	24.0	24.0	935.0	1132.0	64775.0	0.0	0.0	0.0	0.0	0.0	0.0	3123.0
I-FACILITY	0.0	0.0	0.0	0.0	0.0	0.0	0.0	40.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0
I-GPE	0.0	0.0	0.0	0.0	0.0	2.0	2.0	200.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0
I-GSP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
I-LOCATION	0.0	0.0	0.0	0.0	0.0	0.0	0.0	22.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
I-ORGANIZATION	0.0	0.0	0.0	0.0	0.0	2.0	0.0	529.0	0.0	0.0	0.0	0.0	171.0	0.0	1.0
I-PERSON	0.0	0.0	0.0	0.0	0.0	0.0	4.0	856.0	0.0	0.0	0.0	0.0	0.0	162.0	0.0
END	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

شکل 19 : ماتریس Transition (تعداد)

	0	1	2	3	4	5	6	7	8	9	...	2539	2540	2541	2542	2543	2544	2545	2546	2547	2548
START	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
B-FACILITY	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	4.0
B-GPE	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	384.0
B-GSP	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	13.0
B-LOCATION	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	4.0
B-ORGANIZATION	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	2.0	0.0	0.0	0.0	459.0
B-PERSON	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	632.0
O	3817.0	3233.0	3058.0	1854.0	1725.0	1499.0	1286.0	1213.0	873.0	862.0	...	4.0	4.0	4.0	4.0	4.0	0.0	4.0	4.0	4.0	9673.0
I-FACILITY	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	6.0
I-GPE	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	25.0
I-GSP	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
I-LOCATION	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	10.0
I-ORGANIZATION	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	161.0
I-PERSON	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	1.0	0.0	0.0	0.0	522.0
END	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

15 rows × 2549 columns

شکل 19 : ماتریس Emossion (تعداد)

*فرق اساس این سوال با سوال قبل در این است که ترتیب Tag های پیشبینی شده برای یک جمله اهمیت پیدا می‌کند در حالیکه در مساله POS، این امر اهمیتی نداشت.

*برای مثال شروع یک جمله نمی‌توان با Tag هایی که با حرف "I" شروع می‌شود باشد.

*برای Tag های از یک نوع، Tag های از نوع "I" باید همیشه بعد از Tag های با نوع "B" بیایند.

**ولی نکته مهم این است که در Transition matrix شکل 18، تعداد رخداد این وقایع برابر صفر می‌باشد پس طبیعتا مدل اصلا نباید این الگو ها را پیش‌بینی کند پس به نظر من نیاز به تغییری در کد نیست ولی توجه به این نکته حائز اهمیت است.

پ)

حال مانند قبل، دقت مدل را بر روی دادگان صحت بدست می‌آوریم. ولی این دفعه بهترین زوج پارامتر را از روی معیار F1 انتخاب می‌کنیم :

alpha=0.03, beta=3.0, overal_f1=0.3322033898305085
alpha=0.03, beta=4.5, overal_f1=0.3257372654155496
alpha=0.03, beta=6.0, overal_f1=0.32797858099062915
alpha=0.045, beta=3.0, overal_f1=0.3317535545023697
alpha=0.045, beta=4.5, overal_f1=0.32685867381111855
alpha=0.045, beta=6.0, overal_f1=0.3272971160295104
alpha=0.06, beta=3.0, overal_f1=0.3255503669112742
alpha=0.06, beta=4.5, overal_f1=0.32530120481927716
alpha=0.06, beta=6.0, overal_f1=0.3241795043536504

و حالا بهترین زوج پارامتر را استخراج می‌کنیم :

```
AC.argmax() #=> alpha = 0.03 , beta = 3
```

و در نهایت معیار های خواسته شده را بر روی دادگان آزمون بدست می‌آوریم :

alpha=0.03, beta=3, overal_accuracy=0.8991695163654128 , overal_precision=0.28440366972477066 ,
overall recall = 0.4343257443082312 , overall F1-score = 0.3437283437283437

ت)

بله استفاده می‌شود. چالش اصلی این است که هر sequence به عنوان خروجی قابل قبول نیست و باید به سری قوانین حفظ شود. در جدیدترین مقاله ها از روش های **Bidirectional LSTM-CRF Models** ، **Bidirectional LSTM-CNNs** و **Bi-directional LSTM-CNNs-CRF** می‌توان یاد کرد.