دانشگاه تهران

دانشکده علوم و فنون نوین

Sentiment Analysis in Text with LSTM

نام و نام خانوادگی	رامتین کبیری

فهرست

سوال 1 – شبکه های عصبی بزرگ

GPT-3 چیست؟

GPT-3 جدیدترین مدل یادگیری زبانِ ساخته شده توسط گروه نرم‌افزاری OpenAI است. این گروه در ماه مِی ۲۰۲۰ مقاله‌ی مربوط به این مدل یادگیری را منتشر کرد و در ژوئیه همان سال این مدل از طریق واسط برنامه نویسی کاربردی در اختیار تعدادی از آزمونگرهای بتا قرار گرفت. تا کنون این مدل برای نوشتن شعر، خلق داستان‌های ماجراجویانه، یا ساخت سریع و آسان اپلیکیشن‌های ساده مورد استفاده قرارگرفته‌است. افرادی که طی سال‌های اخیر پیشرفت‌های یادگیری ماشین را دنبال کرده‌اند، به خوبی می‌دانند که تمام کارهای فوق قبلاً توسط دیگر سیستم‌های هوش مصنوعی اجرا می‌شدند، پس وجه تمایز GPT-3 چیست؟

GPT-3 افزایش کارایی که از استفاده یک مدل بزرگ‌تر می‌آید را نشان می‌دهد و دنباله‌رو افزایش عظیم در مدل و اندازه اطلاعات است که جدیدترین پیشرفت‌های NLP را توصیف می‌کند. پیام اصلی این بیانیه بیش‌تر از این که در مورد کارایی این مدل در بنچمارک‌ها باشد، در مورد این کشف بوده که به خاطر مقیاسش، GPT-3 قادر است تا وظایفی در NLP که تاکنون با آن‌ها روبه‌رو نشده است را بعد از یک بار دیدن و یا تعداد کمی مثال حل کند. این مساله در تضاد با چیزی است که امروزه انجام می‌شود؛ این که مدل‌ها برای دستورات جدید باید با حجم عظیمی از اطلاعات تمرین داده شوند.

در گذشته نه چندان دور، علی‌رغم پیشرفت‌های حاصل در حوزه RNN، مدل‌های مبتنی بر شبکه‌های عصبی بازگشتی همچنان در نوشتن متن‌های منسجم، ضعیف عمل می‌کردند. متن‌های خروجی مربوط به این دوران بیشتر شبیه پراکنده‌گویی‌های مبهی بودند که به طور آگاهانه به رشته تحریر درآمده بودند. به عبارت دیگر، اکثر آن‌ها به لحاظ قواعد دستورزبان صحیح ولی فاقد انسجام معنایی بودند.

سال ۲۰۱۷ شاهد تحول دیگری در فناوری پردازش زبان طبیعی بودیم. در کنفرانسِ سیستم‌های پردازش اطلاعات عصبی ، مقاله‌ای با عنوان «توجه تنها چیزی است که نیاز دارید» به طور مشترک توسط تیمی از Google Brain و تیمی از پژوهشگران دانشگاه تورنتو ارائه گردید و معماری مبدل را به دنیای فناوری معرفی نمود. اهمیت این معماری جدید از این جهت بود که امکان ایجاد شبکه‌های عصبی بسیار عمیق‌تری را فراهم می‌نمود. شبکه‌های عصبی عمیق از قبل توانمندی خود را در ایجاد انتزاع‌هایی غنی‌تر در حوزه بینایی کامپیوتر نشان داده‌بودند. اکنون این توانمندی در اختیار پژوهشگران حوزه پردازش زبان طبیعی نیز قرار داشت.

قابلیت معماری مبدل در هماهنگی با شبکه‌های عصبی عمیق‌تر این امکان را فراهم کرد تا تیم‌های فناوری، مدل‌هایی بزرگ‌تر از قبل ایجاد کنند. مثلاً، BERT-base، محصول شرکت گوگل، ۱۱۰ میلیون پارامتر دارد؛ BERT-large در زمان ورد به بازار، با ۳۴۰ پارامتر بهترین رکورد عملکرد را ثبت کرد؛ یا CTRL، محصول شرکت Salesforce، در واقع مدلی عظیم با ۱/۶ میلیارد پارامتر است.

اکثر موارد مطرح شده‌ی بالا یا مدل زبانی خودهمبسته هستند یا مدل دارای ماسک. مدل خودهمبسته به مدلی گفته می‌شود که در آن با در نظر گرفتن جمله‌ای، مدل سعی می‌کند واژه بعدی را پیش‌بینی کند. و در مدل دارای ماسک، در صورتی که واژه یا نشانه‌ای در یک جمله دارای «ماسک» باشد، مدل سعی می‌کند آن واژه را پیش‌بینی کند. این رویکرد مناسب شیوه‌های خود نظارتی است. در این روش، مدل به برچسب گذاری دستی نیازی ندارد؛ در ضمن مدل می‌تواند با هر متنی آموزش ببیند. این قابلیت، امکان آموزش با پیکره‌های اطلاعاتی بیشمار و حتی آموزش با کل فضای اینترنت را فراهم نمود.

با همه این اوصاف، مدل‌های مبدلی هزینه‌بر هستند. تعداد پارامترهایی که بر داده‌ها پیاده می‌شوند بسیار زیاد است و در نتیجه فرآیند آموزش بسیار کند پیش می‌رود. از طرفی، محققان به حجم عظیمی از قدرت محاسبات ابری بر روی زیرساخت‌های مدرن و پیشرفته نیاز دارند. لذا مدل‌های جدید تنها توسط بزرگترین تیم‌های فناوری دنیا با بیشترین بودجه عرضه می‌گردد. حتی آموزش مسائل ساده و تنظیم دقیق آن‌ها مستلزم هزاران و شاید ده‌ها هزار نمونه و کامپیوترهای قدرتمند دارای واحد پردازش گرافیکی است.

در چنین شرایطی سری‌های GPT، GPT-2 و GPT-3 مدل‌هایی بسیار معمولی محسوب می‌شوند. این مدل‌ها که توسط شرکت OpenAI معرفی شدند، تا کنون هیچ نوآوری پیشرفته‌ای عرضه نکرده‌اند و تنها وجه تمایز آن‌ها مقیاس است؛ GPT همانند BERT-base دارای ۱۱۰ پارامتراست. GPT-2 در بزرگترین بازنویسی خود ۱,۶ میلیون پارامتر داشته‌است. این مدل چنان در تولید متن‌های منسجم موفق عمل کرد که شرکت OpenAI در ابتدای امر با ابراز نگرانی درباره‌ی اینکه درصورتِ دسترسی عوامل بدخواه به مدل، گسترش اخبار جعلی محتمل خواهد بود حاضر نبود وزن‌های مدل را دراختیار عموم قرار دهد. تعداد پارامترهای GPT-3 بهت آور بود، ۱۷۵ میلیادر پارامتر. برای درک بهتر این شاهکار مهندسی، آزمایشگاه Lambda Labs برآورد کرد که اجرای تنها و تنها یک آموزش بر روی ارزان ترین فضای ابری پردازش گرافیکی موجود در بازار ۳۵۵ سال طول می کشد و ۴,۵ میلیون دلار هزینه دارد.

سوال ۲ – روش های دیگر یادگیری

الف) progressive learning :

در یادگیری مستمر سیستم یادگیرنده توانایی حل کردن یک وظیفه جدید را براساس دانشی که از پیش در فرآیند یادگیری بدست آمده را دارد و همچنین می تواند وظایف قبلی را بدون اینکه بر دانش مورد نیازش تاثیر منفی بگذارد انجام دهد. یادگیری مستمر کلید پیشرفت در یادگیری ماشین و هوش مصنوعی می باشد.

یادگیری پیشرو (progressive learning) یک چارچوب یادگیری عمیق برای یادگیری مستمر می باشد که سه رویه را دنبال می کند که شامل: برنامه آموزشی (curriculum)، پیشروی (progression) و هرس کردن(pruning).

روند برنامه آموزشی از انتخاب فعال یک وظیفه از میان مجموعه وظایف کاندید برای یادگیری استفاده می کند.

روند پیشروی برای افزایش ظرفیت مدل از اضافه کردن پارامترهای جدید استفاده می کند تا پارامترهای آموخته شده در وظایف قبلی را تحت تاثیر قرار دهد و این در حالی اتفاق می افتد که در حال یادگیری از داده های در دسترس مربوط به وظیفه جدید -بدون اینکه مستعد فراموشی فاجعه بار باشد- می باشد.

روند هرس کردن برای جلوگیری از افزایش بی رویه پارامترها با اضافه شدن وظایف جدید است چرا که برخی پارامترهای نامرتبط می توانند بر دانش بدست آمده و عملکرد اثر منفی بگذارند.

یادگیری پیشرو (progressive learning) بر روی برخی مسائل دسته بندی (classification) نظارت شده (supervised) مانند تشخیص تصویر (image recognition) و تشخیص گفتار (speech recognition) مورد ارزیابی قرار گرفته شده است که نتایج این ارزیابی نشان می دهد که یادگیری پیشرو در این موارد عملکرد بهتری نسبت به روش های پایه دارد .

وقتی که وظایف به هم مرتبط باشند ، روش یادگیری پیشرو با سرعت بیشتری یادگیری را انجام میدهد و برای رسیدن به یک عملکرد تعمیم یافته مطلوب نیاز به پارامترهای اختصاصی کمتر دارد.

ب) Meta Learning :

فرا یادگیری زیرشاخه‌ای از یادگیری ماشین است که در آن الگوریتم‌های یادگیری خودکار بر فراداده ها برای انجام آزمایش‌های یادگیری ماشین اعمال می‌شوند.

هدف اصلی فرایادگیری به صورت زیر بیان شده‌است: درک اینکه چگونه یادگیری خودکار می‌تواند در حل مشکلات یادگیری انعطاف‌پذیر شود تا منجر به بهبود عملکرد الگوریتم‌های یادگیری موجود یا یادگیری خود الگوریتم یادگیری شود.

یادگیری فقط درصورتی به درستی انجام می‌شود که سوگیری با مسئله یادگیری مورد نظر مطابقت داشته باشد. یک الگوریتم یادگیری ممکن است در یک حوزه عملکرد بسیار خوبی داشته باشد، اما در حوزه بعدی چنین نباشد. این امر محدودیت‌های شدیدی را در استفاده از تکنیک‌های یادگیری ماشین یا داده کاوی تحمیل می‌کند.

با استفاده از انواع مختلف فراداده، مانند خصوصیات مسئله یادگیری، خصوصیات الگوریتم (مانند معیارهای عملکرد)، یا الگوهایی که قبلاً از داده‌ها کشف شده‌است، می‌توان الگوریتم‌های یادگیری مختلف را آموخت، انتخاب کرد، تغییر داد، یا ترکیب کرد تا به‌طور مؤثر یک مسئله یادگیری را حل کرد.

برخی از رویکردهایی که به عنوان مواردی از فرا یادگیری مشاهده شده‌اند، عبارت اند از:

- ✓ در سال ۱۹۹۳، یورگن اشمیدوبر نشان داد که چگونه RNNهای «خود ارجاع» می‌توانند با استفاده از پس انتشار در اصل یاد بگیرند که الگوریتم تغییر وزن خود را اجرا کنند، که ممکن است کاملاً متفاوت از خود پس انتشار باشد.
- ✓ نوع افراطی فرا یادگیری تقویتی توسط ماشینGödel ، یک ساختار نظری که می‌تواند هر بخشی از نرم‌افزار خود (شامل یک اثبات‌گر قضیه عمومی) را بازرسی و اصلاح کند، تجسم یافته‌است. این نوع فرا یادگیری می‌تواند به خود ارتقا بخشی بازگشتی به روشی که بهینه بودن آن قابل اثبات است برسد.
- ✓ یادگیری ماشین اتوماتیک مانند پروژه "AI building AI" گوگل‌برین، که به نقل از گوگل برای مدت کوتاهی از بنچمارک‌های موجود در سال ۲۰۱۷ از ایمیجنت فراتر رفت.

ج) zero shot learning :

یادگیری بدون نمونه (zero shot learning) یک مشکل در یادگیری ماشینی است. این مشکل وقتی است که در زمان آزمون، یک یادگیرنده نمونه‌هایی از کلاس‌هایی را که در طول آموزش مشاهده نکرده است را مشاهده می‌کند و باید کلاس مربوط به آنها را پیش‌بینی کند.

روشهای یادگیری بدون نمونه معمولاً با تداعی کلاسهای مشاهده شده و مشاهده نشده از طریق نوعی اطلاعات کمکی، که خصوصیات تمایز قابل مشاهده اشیا را رمزگذاری می‌کند، کار می‌کنند. این مشکل به‌طور گسترده‌ای در بینایی رایانه ، پردازش زبان طبیعی و درک ماشین مورد مطالعه قرار گرفته‌است.

یادگیری بدون نمونه در زمینه‌های : طبقه‌بندی تصویر ، تقسیم‌بندی معنایی ، تولید تصویر ، تشخیص شی و پردازش زبان طبیعی اعمال شده است.

د) one shot learning :

یادگیری تک شات(One-shot learning) ، یک مسئله طبقه بندی اشیا، بیشتر در مسائل بینایی ماشین، می باشد. در حالی که اکثر الگوریتم‌های دسته‌بندی مبتنی بر یادگیری ماشین، برای آموزش صدها یا هزاران مثال نیاز دارند، هدف یادگیری One Shot ، طبقه‌بندی اشیا از یک یا تنها چند مثال است. اصطلاح یادگیری Few-shot learning نیز برای این مسائل استفاده می شود، به خصوص زمانی که به بیش از یک مثال نیاز باشد.

یادگیری تک‌شات شامل سه چالش اصلی است:

- ✓ نمایش: اشیاء و دسته بندی ها چگونه باید توصیف شوند؟
- ✓ یادگیری: چگونه می توان چنین توصیفاتی ایجاد کرد؟
- ✓ تشخیص: چگونه می توان یک شی شناخته شده را بدون در نظر گرفتن نقطه دید، نور و غیره فیلتر کرد؟

یادگیری تک شات با تاکید بر انتقال دانش، که از مقوله‌های قبلاً آموخته شده استفاده می‌کند، با الگوریتم‌های تشخیص تک شی و تشخیص دسته استاندارد متفاوت است.

یادگیری تک شات بطور عملی در زمینه تشخیص چهره مورد استفاده قرار گرفته است. برای مثال: تایید و شناسایی چهره زمانی که اشخاص باید بطور دقیق با وجود شرایط ظاهری متفاوت، نور ، مدل مو ، کلاه و ... دسته بندی (classify) شوند آن هم تنها با داشتن یک یا دو نمونه قالب عکس ، کاری که برای شناسایی گذرنامه (Passport identification) در فرودگاه ها انجام می گیرد.

در یادگیری تک شات از CNN (convolutional neural network) استفاده می شود.

محدودیت یادگیری تک شات : در حالی که یادگیری تک شات نیاز به آموزش تعداد بسیار زیادی تصویر برای یک مدل را برطرف می کند اما موارد استفاده این روش بسیار خاص می باشد. شما نمی توانید از مدلی که برای تشخیص شباهت تصویر گذرنامه و شخصی که به دوربین نگاه می کند برای تشخیص وجود گربه یا ماشین در دو تصویر استفاده کنید.

سوال3 – پیاده سازی: تحلیل احساس در متن با شبکه LSTM

پیاده سازی در 3 فاز به شرح ذیل انجام گرفته است:

تهیه و تنظیم dataset :

```
import os

os.chdir("aclImdb")
os.chdir("train")
os.chdir("neg")
print(os.getcwd())
directorylist = os.listdir()
#print(directorylist)
datalist = []
for i in directorylist:
    file = open(i)
    data = file.readlines()
    file.close()
    my_str = data[0].lower()
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''

    no_punct = ""
    for char in my_str:
        if char not in punctuations:
            no_punct = no_punct + char

    datalist.append(no_punct)

print("\n",datalist[3])

wordseperatedataset = []
for i in datalist:
```

```
    words = i.split(" ")
    wordseperatedataset.append(words)

print("\n",wordseperatedataset[3])


import csv

os.chdir("../")
os.chdir("../")
os.chdir("../")
file = open("train_neg.csv", "w", newline="")
writer = csv.writer(file)
writer.writerows( wordseperatedataset )

file.close()
```

کد بالا نمونه ای برای واحد بندی و ساخت لیست از روی دیتاها می باشد .

در کدها برای تنظیم داده ها (جملات) ابتدا تمام آن ها را به حروف کوچک و سپس تمام علائم را از آن حذف می کنیم سپس هر جمله را با کمک split بر مبنای کارکتر space جدا می کنیم و حالا جمله تبدیل به یک لیست از کلمات می شود.

این کار 5 بار انجام شده است چرا که 5 تا فولدر در فولدر دیتاست aclImdb قرار دارد که به شرح زیر است:

1. Train :
   1.1 neg
   1.2 pos
   1.3 unsup
2. test:
   2.1 neg
   2.2 pos

در فولدر "واحد بندی و تهیه لیست" کدهای انجام این قسمت قرار دارد.

لیست های ساخته و پرداخته شده توسط این کدها به صورت فایل csv ذخیره شده و داخل فولدر "دیتاست تجمیع شده و واحد بندی شده" قرار گرفته است.

سپس این فایل های csv داخل google drive ، upload گشته و در google colab مورد استفاده قرار گرفته است که فراخوانی این فایل ها بصورت زیر است:

```
# فراخوانی داده‌های آموزش
from google.colab import drive
drive.mount('/content/drive')

train_file_neg= open('/content/drive/MyDrive/HW4_dataset/train_neg.csv', 'r')
train_list_neg = train_file_neg.readlines()
train_file_neg.close()

train_file_pos= open('/content/drive/MyDrive/HW4_dataset/train_pos.csv', 'r')
train_list_pos = train_file_pos.readlines()
train_file_pos.close()

train_file_unsop= open('/content/drive/MyDrive/HW4_dataset/train_unsop.csv', 'r')
train_list_unsop = train_file_unsop.readlines()
train_file_unsop.close()

# فراخوانی داده‌های آزمایش
test_file_neg= open('/content/drive/MyDrive/HW4_dataset/test_neg.csv', 'r')
test_list_neg = test_file_neg.readlines()
test_file_neg.close()

test_file_pos= open('/content/drive/MyDrive/HW4_dataset/test_pos.csv', 'r')
test_list_pos = test_file_pos.readlines()
test_file_pos.close()
```

```
Mounted at /content/drive
```

BERT :

برای نوشتن تابع تبدیل کلمه به بردار کتابخانه های زیر import شده اند.

```
[4]  import tensorflow as tf
     import tensorflow_text as text
     import tensorflow_hub as hub
     print(tf.__version__)
```

```
2.11.0
```

```
[5]  bert_preprocessor = hub.KerasLayer(
         "https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
```

```
WARNING:tensorflow:Please fix your imports. Module tensorflow.python.t
```

```
[14] bert_encoder = hub.KerasLayer(
         "https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

تابع تبدیل کلمه به بردار با استفاده از preprocessor و encoder ، BERT انجام شده که این تابع به شکل زیر است:

```
def get_sentence_embeding(senteces):
  preprocessed_text = bert_preprocessor(senteces)
  return bert_encoder(preprocessed_text)['pooled_output']

get_sentence_embeding(["500$ discount. hurry up",
                       "bhavin, are you up for a game tomorrow?"
                       ])
```

```
<tf.Tensor: shape=(2, 768), dtype=float32, numpy=
array([[-0.8435169 , -0.51327264, -0.88845724, ..., -0.74748874,
        -0.75314736,  0.91964483],
       [-0.8765555 , -0.50088423, -0.95878726, ..., -0.8938003 ,
        -0.71859884,  0.8784346 ]], dtype=float32)>
```

یک نمونه برای مشاهده عملکرد این الگوریتم همراه با خروجی هم در تصویر بالا قابل مشاهده است.

روی یکی از نمونه های (نظرات) دیتاست مسئله نیز این تابع مورد بررسی قرار گرفته و کد و خروجی آن به شرح زیر است:

```
print("test_list_pos[1]: ",test_list_pos[1])
a = test_list_pos[1].split(',')
b = get_sentence_embeding(a)
print("bert_vec: ",b)
```

```
test_list_pos[1]:  actor,turned,director,bill,paxton,follows,up,his,

bert_vec:  tf.Tensor(
[[-0.9374246  -0.6122873  -0.82031155 ... -0.52665734 -0.6588041
   0.8915311 ]
 [-0.87391037 -0.14990044  0.3370829  ...  0.25290614 -0.5821726
   0.8747224 ]
 [-0.9359918  -0.4552164  -0.6063657  ... -0.30570996 -0.6558526
   0.87928855]
 ...
 [-0.87888384 -0.20548141  0.47031605 ...  0.4175205  -0.5982299
   0.89078474]
 [-0.8039575  -0.16494808  0.48876008 ...  0.28318304 -0.5206983
   0.83352554]
 [-0.8164539  -0.21256424  0.38060272 ...  0.35062334 -0.5337261
   0.8098593 ]], shape=(344, 768), dtype=float32)
```

LSTM :

ابتدا کتابخانه های لازم برای پیاده سازی LSTM را فراخوانی می کنیم:

```
[ ]  from tensorflow import keras
     from keras.layers import Embedding, Dense, LSTM
     from keras.models import Sequential, load_model
     from keras.losses import BinaryCrossentropy
     from keras.optimizers import Adam
     from keras.preprocessing import sequence
```

```
[ ]  from keras.datasets import imdb
     from keras.utils import pad_sequences
```

برای بهبود سرعت training دستورات زیر به کار رفته است :

```
# speed up
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
# Disable eager execution
tf.compat.v1.disable_eager_execution()
```

برای راحتی کار با google colab و عدم وابستگی به داده های داخل google drive از دیتاست IMDB آماده موجود در کتابخانه tensorflow استفاده شده است.

```
# Load dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_distinct_words)
print(x_train.shape)
print(x_test.shape)

# Pad all sequences
padded_inputs = pad_sequences(x_train, maxlen=max_sequence_length, value = 0.0) # 0.0 because it corresponds with <PAD>
padded_inputs_test = pad_sequences(x_test, maxlen=max_sequence_length, value = 0.0) # 0.0 because it corresponds with <PAD>
```

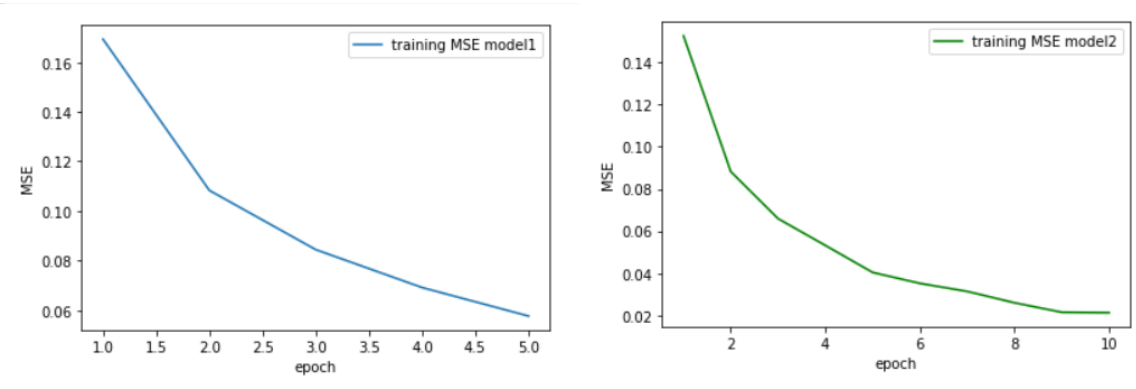
حالا نوبت به config مدل می رسد.

```
# Model configuration
additional_metrics = ['accuracy', 'mean_squared_error']
batch_size = 50
embedding_output_dims = 64
loss_function = BinaryCrossentropy()
max_sequence_length = 300
num_distinct_words = 5000
number_of_epochs = 5
optimizer = Adam()
validation_split = 0.20
verbosity_mode = 1
# Define the Keras model
model = Sequential()
model.add(Embedding(num_distinct_words, embedding_output_dims, input_length=max_sequence_length))
model.add(LSTM(10))
model.add(Dense(1, activation='sigmoid'))
# Compile the model
model.compile(optimizer=optimizer, loss=loss_function, metrics=additional_metrics)
```

دو مدل ابتدایی و ساده که با عنوان : model1 و model2 در فولدر "init_model" وجود دارند شامل کد و خروجی ها و نتایج این مدل هاست.

تنها تفاوت این دو مدل در تعداد epoch های آن هاست که مدل اول epoch=5 و در مدل دوم epoch=10 است.

باتوجه به Accuracy و نمودار تغییرات MSE با افزایش epoch روند Accuracy افزایشی و روند MSE (همانطور که درنمودار زیر مشاهده می کنید) کاهشی می باشد اما Accuracy در داده های آزمایش با افزایش epoch کاهش یافت. بنظر می رسد که افزایش epoch از 5 به 10 باعث نزدیک شدن به overfit می شود درنتیجه دقت lstm روی نمونه های آزمایش کاهش یافته است.



نتیجه Accuracy در مدل 1 و 2 در فولدر "init_model" وجود دارد که این نتیجه برای model1 (دارای epoch 5) Accuracy: 85.29199957847595% و برای model2 (دارای epoch 10) Accuracy: 80.03600239753723% می باشد.

Bidirectional LSTM

برای دوطرفه کردن LSTM کافی است کتابخانه Bidirectional را به شکل زیر فراخوانی کنیم

```
from keras.layers import Bidirectional
```

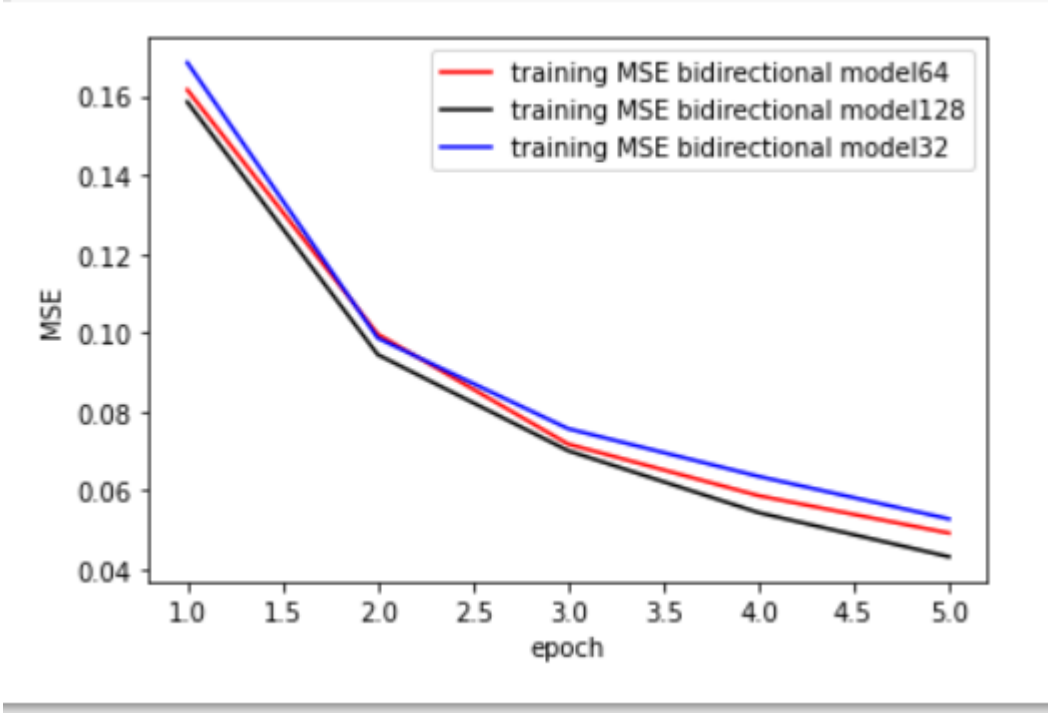
و سپس تنها تغییری که لازم است انجام دهیم تبدیل خط(1) به خط(2) است

```
model.add(LSTM(10))   :(1)خط

model.add(Bidirectional(LSTM(10)))   :(2)خط
```

این الگوریتم (Bidirectional LSTM) را با سه تعداد مختلف واحد های حافظه این الگوریتم را اجرا کردیم و تمام قسمت های این 3 کد و نتایج و اجراهایش در فولدر "bidirectional_model" قرار دارد.

یک مقایسه بر روی نمودار MSEهای این سه مورد انجام گرفته کر در شکل زیر قابل مشاهده می باشد.



از نمودار بالا اینطور استنباط می شود که با افزایش تعداد واحدهای حافظه سریع تر میزان MSE کاهش می یابد و نتیاجتا میزان دقت بر روی فرآیند آموزش سریع تر افزایش می یابد .

مرجع‌ها

1. https://hooshio.com - در دنیا آشنا شویم AGI اولین هوش مصنوعی عمومی GPT-3 با
2. https://fanology.ir - openAI را بشناسید چیست؟ نسخه‌ی جدید پردازش زبان طبیعی
3. Progressive learning: A deep learning framework for continual learning

   Authors: Haytham M Fayek , Lawrence Cavedon , Hong Ren Wu
4. Wikipedia
5. https://iq.opengenus.org - one-shot learning in ML
6. اوپن مایند : https://open-mind.ir

   تعاریف و تفاوت های L2 Regularization و L1 Regularization در یادگیری ماشین

   Author: حامد آهنگری