به نام خدا

ملیکا احمدی رنجبر 97521036

1)

a. Convolution1: 24 * 24 * 64
b. MaxPooling1: 12 * 12 * 64
c. Convolution2: 8 * 8 * 128
d. MaxPooling2: 4 * 4 * 128
e. Convolution3: 2 * 2 * 256
f. MaxPooling3: 1 * 1 * 256
g. Flatten: 256

ابعاد خروجی هر لایه Dense برابر با تعداد ورودی‌های آن است. بنابراین اگر لایه‌ای 128 Unit دارد، 128 ورودی خواهد داشت، و تعداد پارامترهای لایه Denseنیز متناسب با تعداد Unitهای پیش از آن و لایه قبل متصل به آن است و تعداد Unitهای همان لایه. البته دوباره به علاوه تعداد لایه های قبل نیز می شود به دلیل وجود Bias.

MaxPooling و Flatten پارامتر ندارند و ابعاد خروجی آن‌ها نیز نوشته شده‌است. Concatenate نیز خروجی های موجود را Merge می کند. بنابراین پارامتر خاصی نخواهد داشت و خروجی آن صرفا جمع ابعاد ورودی آن است که در اینجا برابر 256 است. برای RNN نیز توجه به ابعاد ورودی برای محاسبه خروجی اهمیت زیادی دارد. از طرفی تعداد پارامترها برای Convolution Layer ها نیز به شکل زیر حساب خواهد شد:

(تعداد فیلتر لایه قبل * KernelSize * KernelSize + 1) * تعداد فیلتر لایه دوم

A -> (7 * 7 * 1 + 1) * 64 = 3200, C -> (5 * 5 * 64 + 1) * 128 = 204928

E -> (3 * 3 * 128 + 1) * 256 = 295168, First Dense -> (256 * 1 * 1 + 1) * 128 = 32896

این مراحل و پارامترها برای قسمت Convolutional بود. به عنوان مثال برای بعد از Concatenate خواهیم داشت:

(256 + 1) * 128 = 32986, (128 + 1) * 1000 = 129000

پس از قسمت، وارد شبکه RNN شده و باید پارامترهای مربوط آن را محاسبه کنیم.

ولی تا به قبل آن تعداد پارامترها برابر = 698088 پس از این نیز با توجه به RNN به آن اضافه خواهد شد.

2) در این سوال ابتدا باید Dataset لازم برای ورودی دادن را آماده می‌کردیم که با توجه به مراحل توضیح داده‌شده در Document جلو رفت:

همانطور که واضح است، ابتدا تمام Fileهایی که شامل اخبار Crude بودند را پیدا نموده، سپس 100 تای اول آن‌ها را نگه می‌داریم، در در هر کدام کلمات موجود در آن را یک خبر می‌کنیم (با استفاده از Join).

بعد از این مرحله آماده برای استفاده از توضیحات و Code آماده‌ای که در اختیار ما قرار داده‌شده هستیم.

Cellهای موجو بعد از این تماما Codeهای آماده سایت اعلام شده‌است و تنها در صورت نیاز تغییرات لازم روی آن‌ها اعمال شده‌است.

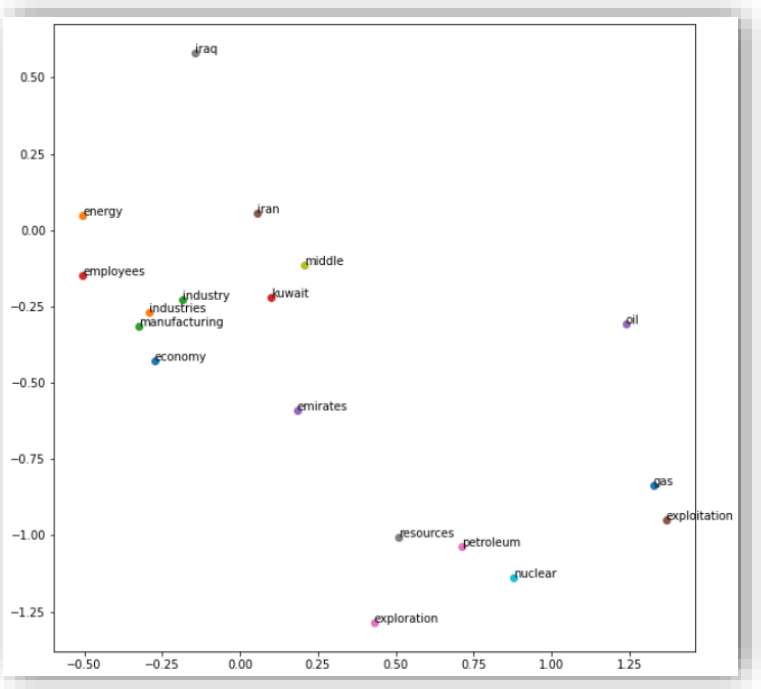
نتیجه حاصل از Training اهمیت خاصی ندارد و ما تنها نیاز به وزن‌های آن داریم. پس از اتمام فرایند آموزش، نمودار پراکندگی کلمات داده شده در

```
# Create Input
# Crude
input = []
for field in reuters.fileids('crude'):
    input.append(field)
# The first One Hundred Files
input = input[0:100]

texts = []
for files in input:
    file_words = reuters.words(files)
    output = " ".join(file_words)
    texts.append(output)

print(len(texts))
```

```
100
```



Document را رسم کرده، که به شکل زیر است:

پس از این کار، یک کلمه Iran را برای آنکه 5 کلمه مشابه آن را از نظر کمترین فاصله ممکن پیدا کند به تابع Find Similar می‌دهیم که نتیجه به صورت زیر است (فاصله آن‌ها را نیز تا کلمه مد نظر ما برمی‌گرداند):

چند مورد دیگر نیز پس از این مورد انجام شد و نتایج آن در Notebook موجود است.

a. پیش‌پردازش‌هایی که بر روی متون داده شده صورت گرفته است بیشتر به منظور حذف برخی از کلمات موجود در آن است که Stop Words نام دارند. کلماتی همچون The, A, Is و ... . و بنابراین از این طریق کلمات کلیدی‌تر را در نظر گرفته و برای آن‌ها مراحل یاد شده را انجام می‌دهیم. حتی کلماتی مانند صفات نیز گاها ممکن است به عنوان Stop Words باشند.

```
# Finad Similarities
similarities = find_similar('iran', embedding_dict, 5)
print(similarities)
```

```
[('nations', 0.004070975), ('deposits', 0.01005239), ('marketer', 0.03207974), ('seven', 0.04138999), ('maintain', 0.047143772)]
```

b. داده‌های آموزشی موجود تعداد زیادی File دارد که هر یک از این Fileها دارای تعداد متفاوتی کلمه (Word) می باشد. از طرفی هر کدام از این Fileها یا در دسته Test هستند یا Training. و همچنین هر یک از این Fileها

```
>>> reuters.categories('training/9865')
['barley', 'corn', 'grain', 'wheat']

>>> reuters.categories(['training/9865', 'training/9880'])
['barley', 'corn', 'grain', 'money-fx', 'wheat']

>>> reuters.fileids('barley')
['test/15618', 'test/15649', 'test/15676', 'test/15728', 'test/15871', ...]

>>> reuters.fileids(['barley', 'corn'])
['test/14832', 'test/14858', 'test/15033', 'test/15043', 'test/15106',
'test/15287', 'test/15341', 'test/15618', 'test/15618', 'test/15648', ...]
```

شامل یک یا چند Category مختلف می‌شوند. بنابراین یا می‌توان بر اساس نام File جلو رفت و Categoryهای آن را دید و یا برعکس، بر اساس Category. به مثال زیر توجه کنید:

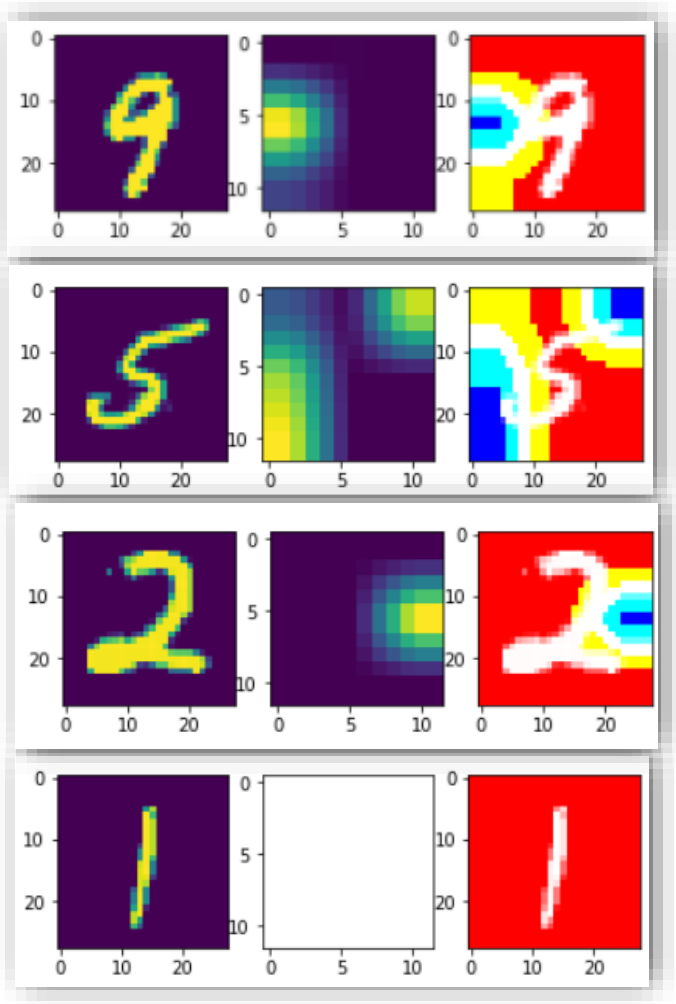
c. تاثیر Window Size در Word Embedding: در صورت افزیش ابعاد Window در حین آموزش، کلمات بیشتری را قبل و بعد از یک واژه بررسی می‌کنیم، که باعث می‌شود نسبت به کلیت مطلب و جمله بیشتر آگاه شود و اطلاعات بیشتری را در رابطه با موضوع بحث و Topic برداشت کند، اما از طرفی نیز ممکن است دقت برای خود کلمه پایین آید. و به طور کلی کلماتی که در آن موضوع خاص مشابه و پرکاربرد هستند را می‌تواند پیدا کند. اما در Window های کوچک‌تر، اطلاعات و آموزش نسبت به خود کلمه بسیار بیشتر می‌شود نه نسبت به محتوا، و کلمات نزدیک واژه مد نظر ما در جمله فقط بررسی می‌شوند، مثلا با فاصله یک یا دو. و اینکه چه کلماتی از نظر معنایی و کارایی مشابه این واژه هستند با Window Size کوچک بهتر است. بنابراین با توجه به سوال پیش رو باید در انتخاب Window Size دقت کنیم، چراکه نقش بسیار کلیدی در این Algorithm دارد.

d. بردارهای Word Embedding نهایی پس از تمام شدن Training با توجه به Weightsهای موجود از Model بدست می‌آیند و در واقع وزن‌های بین لایه ورودی و میانی است که Train می‌شوند و در نهایت به عنوان Word Embedding استفاده می‌شوند.

3) ابتدا موارد ذکر شده را مرحله به مرحله اجرا کرده و خروجی حاصل را به نمایش می‌گذاریم همانطور که در Notebook قابل مشاهده است. پس از آن Training را آغاز مرده و به Accuracy زیر خواهیم رسید:

```
Epoch 10/15
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.0097 - accuracy: 0.9970 - val_loss: 0.0287 - val_accuracy: 0.9916
Epoch 11/15
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.0069 - accuracy: 0.9977 - val_loss: 0.0313 - val_accuracy: 0.9920
Epoch 12/15
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.0081 - accuracy: 0.9973 - val_loss: 0.0357 - val_accuracy: 0.9916
Epoch 13/15
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.0071 - accuracy: 0.9975 - val_loss: 0.0392 - val_accuracy: 0.9904
Epoch 14/15
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.0070 - accuracy: 0.9976 - val_loss: 0.0308 - val_accuracy: 0.9918
Epoch 15/15
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.0050 - accuracy: 0.9982 - val_loss: 0.0281 - val_accuracy: 0.9931
```

که بسیار مطلوب و خوب است، و پس از آموزش، نوبت به اجرای Grad CAM Algorithm می‌رسد. در حقیقت انجام این Algorithm به دلیل نمایش دقیق Featureهایی است که در Classification تاثیرگذار بوده‌است. به این منظور از Grad CAM استفاده می‌کنیم، و در نهایت همان عکس اصلی را به تغییراتی که نمایش‌دهنده قسمت‌های مهم عکس در تصمیم‌گیری Classification بودند را نشان می‌دهیم. اما نکته آن است که تنها برای لایه آخر Convolution این Grad CAM Algorithm اجرا می‌شود. در واقع نمایش می‌دهد که هر مکان و نقطه چقدر در Classification اهمیت داشته است. نتایج حاصل به شکل زیر است:



نمونه‌هایی از این قبیل نیز ممکن است موجود باشد و دلیل آن HeatMap ایجاد شده آن است، که نقطه حساس‌تر و مهم‌تر پیدا نکرده است.

رنگ خروجی متفاوت است اما همان نتیجه را اعلام می‌کند، و این به دلیل رنگ اولیه Input ما است.

4) منابع:

a. https://github.com/jaekookang/mnist-grad-cam/blob/master/gradcam.py
b. https://www.fatalerrors.org/a/obtain-text-corpus-and-vocabulary-resources.html
c. https://testanother-codwar.medium.com/sentiment-analysis-on-nltk-reuters-corpus-2feed2a695e7
d. https://stackoverflow.com/questions/22272370/word2vec-effect-of-window-size-used/30447723
e. https://stackoverflow.com/questions/42786717/how-to-calculate-the-number-of-parameters-for-convolutional-neural-network