Farshad Borjalizade

HW5 – Neural Networks for Sentiment Analysis

Natural Language Processing

University of Tehran

Mordad 1400

گزارش کار انجام شده

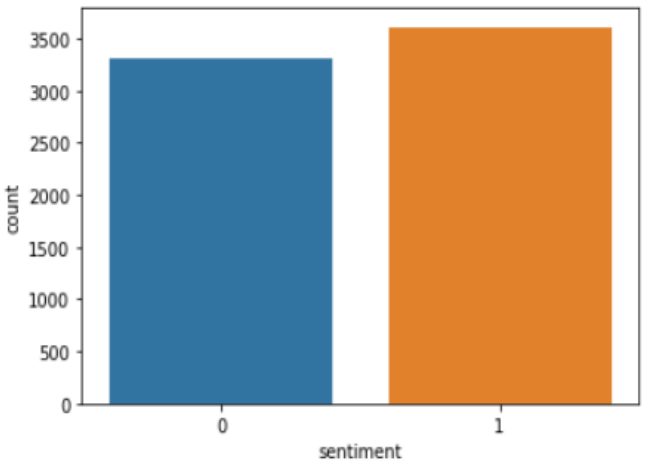
در این تمرین قصد داریم نظرات کاربران در مورد فیلم ها را مورد بررسی قرار دهیم و آنها را به دو دسته نظر منفی و مثبت دسته بندی کنیم. این کار را به دو صورت و با استفاده از شبکه های feedforward و RNN انجام می دهیم.

مجموعه داده

مجموعه داده ای که در این تمرین در اختیار داریم مربوط به نظرات کاربران در مورد فیلم های سینمایی است (movie review dataset of Socher et al. (2013)). هر کدام از نظرات به وسیله صفر و یک برچسب گذاری شده اند که صفر به معنای منفی بودن نظر و یک به معنای مثبت بودن نظر کاربر است، برای درک بهتر داده ها آنها به صورت جدولی و نمودار در تصاویر زیر نشان داده شده است .

	review	sentiment
0	the rock is destined to be the 21st century 's...	1
1	the gorgeously elaborate continuation of `` th...	1
2	singer\/composer bryan adams contributes a sle...	1
3	yet the act is still charming here .\n	1
4	whether or not you 're enlightened by any of d...	1
...	...	...
6915	a real snooze .\n	0
6916	no surprises .\n	0
6917	we 've seen the hippie-turned-yuppie plot befo...	1
6918	her fans walked out muttering words like `` ho...	0
6919	in this case zero .\n	0

6920 rows × 2 columns



پیش پردازش داده ها

به جای اینکه داده ها را به صورت خام به شبکه دهیم برای گرفتن نتیجه بهتر ابتدا باید پیش پردازش را بر روی آنها انجام داد. با تعریف تابع preprocess_text() و حذف کاراکترهای غیر لازم داده ها را به صورت خوبی از کلمات در می آوریم. همانند همه کارهای یادگیری ماشین دیگر اجازه دستکاری داده های test را نداریم به همین خاطر مجموعه داده آموزش را به دو قسمت validation و train با نسبت ۸ به دو تقسیم می کنیم. با استفاده از کلاس Tokenizer از کتابخانه keras ده هزار کلمه ای که بیشترین رخداد را دارند انتخاب می کنیم و این کتابخانه آنها را به صورت دیکشنری ای در می آورد که به ازای هر لغت تعداد تکرار آن را به عنوان مقدار آن کلید در نظر می گیرد و با استفاده از texts_to_sequences لغات را به دنباله ای از اعداد تبدیل می کنیم.

حال برای تشکیل ماتریس embedding کلمات از فایل موجود که به صورت ۳۰۰ تایی کلمات را embed کرده است استفاده می کنیم، هر چند که فایلی که به صورت ۵۰ تایی هم کلمات را embed کرده است موجود است اما برای نتیجه بهتر به طور پیش فرض از طول ۳۰۰ تایی استفاده می کنیم.

آموزش شبکه

آموزش شبکه را به صورت با دو مدل انجام خواهیم داد:

۱. **مدل feedforward** : با استفاده از چندین لایه fully connected و تابع فعال ساز relu و بهینه ساز adam به همراه epochs=3 شبکه را آموزش می دهیم، در تصویر زیر خلاصه ای از مدل را می توان مشاهده کرد.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 1000, 300)         3670200
_________________________________________________________________
dense_9 (Dense)              (None, 1000, 512)         154112
_________________________________________________________________
dense_10 (Dense)             (None, 1000, 256)         131328
_________________________________________________________________
dense_11 (Dense)             (None, 1000, 128)         32896
_________________________________________________________________
dense_12 (Dense)             (None, 1000, 64)          8256
_________________________________________________________________
module_wrapper_1 (ModuleWrap (None, 64000)             0
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 64001
=================================================================
Total params: 4,060,793
Trainable params: 4,060,793
Non-trainable params: 0
```

که برای داده های آموزش و تست به دقت های زیر می رسیم.

```
173/173 [==============================] - 3s 17ms/step - loss: 0.1032 - accuracy: 0.9651
Training Accuracy: 0.9651

28/28 [==============================] - 1s 22ms/step - loss: 0.5629 - accuracy: 0.7993
Testing Accuracy:  0.7993
```

همین مدل را بر روی داده هایی که به صورت ۵۰ تایی embed شده اند را اجرا کردیم که به دقت های زیر رسیدیم.

```
173/173 [==============================] - 2s 9ms/step - loss: 0.2482 - accuracy: 0.9012
Training Accuracy: 0.9012

28/28 [==============================] - 0s 10ms/step - loss: 0.5342 - accuracy: 0.7810
Testing Accuracy:  0.7810
```

۲. مدل Bidirectional LSTM :

در پیاده سازی این مدل بر اساس تجربه ای که داشتم، داده به ساختار مدل بسیار حساس بود و با جابجایی کوچکی از هایپر پارامترها دقت مدل بسیار کم یا زیاد می شد. با آزمون و خطای بسیار با مدل Bidirectional LSTM به نتیجه مطلوب رسیدم و مدل به این شرح است که با استفاده از لایه های LSTM و Bidirectional از کتابخانه keras با تعداد units=64 برای

لایه LSTM و دو لایه fully connected با اندازه های 64 و 1 به همراه dropout=0.5 مدل را ساختم.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 1000, 300)         3670200
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               186880
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 3,865,401
Trainable params: 3,865,401
Non-trainable params: 0
```

برای آموزش مدل هم از بهینه ساز rmsprop به همراه epochs=10 استفاده شده است که نتایج بر روی داده های آموزش و تست در تصویر زیر قابل مشاهده است.

```
173/173 [==============================] - 8s 40ms/step - loss: 0.1516 - accuracy: 0.9429
Training Accuracy: 0.9429

28/28 [==============================] - 1s 40ms/step - loss: 0.4403 - accuracy: 0.8188
Testing Accuracy:  0.8188
```

در جدول زیر خلاصه ای از نتایج حاصل شده را می توان دید.

	Accuracy on Train	Accuracy on Test
Feedforward300	96.5%	79.9%
Feedforward50	90.1%	78.1%
Bidirectional LSTM300	94.3%	81.8%
Bidirectional LSTM50	84.9%	78.3%