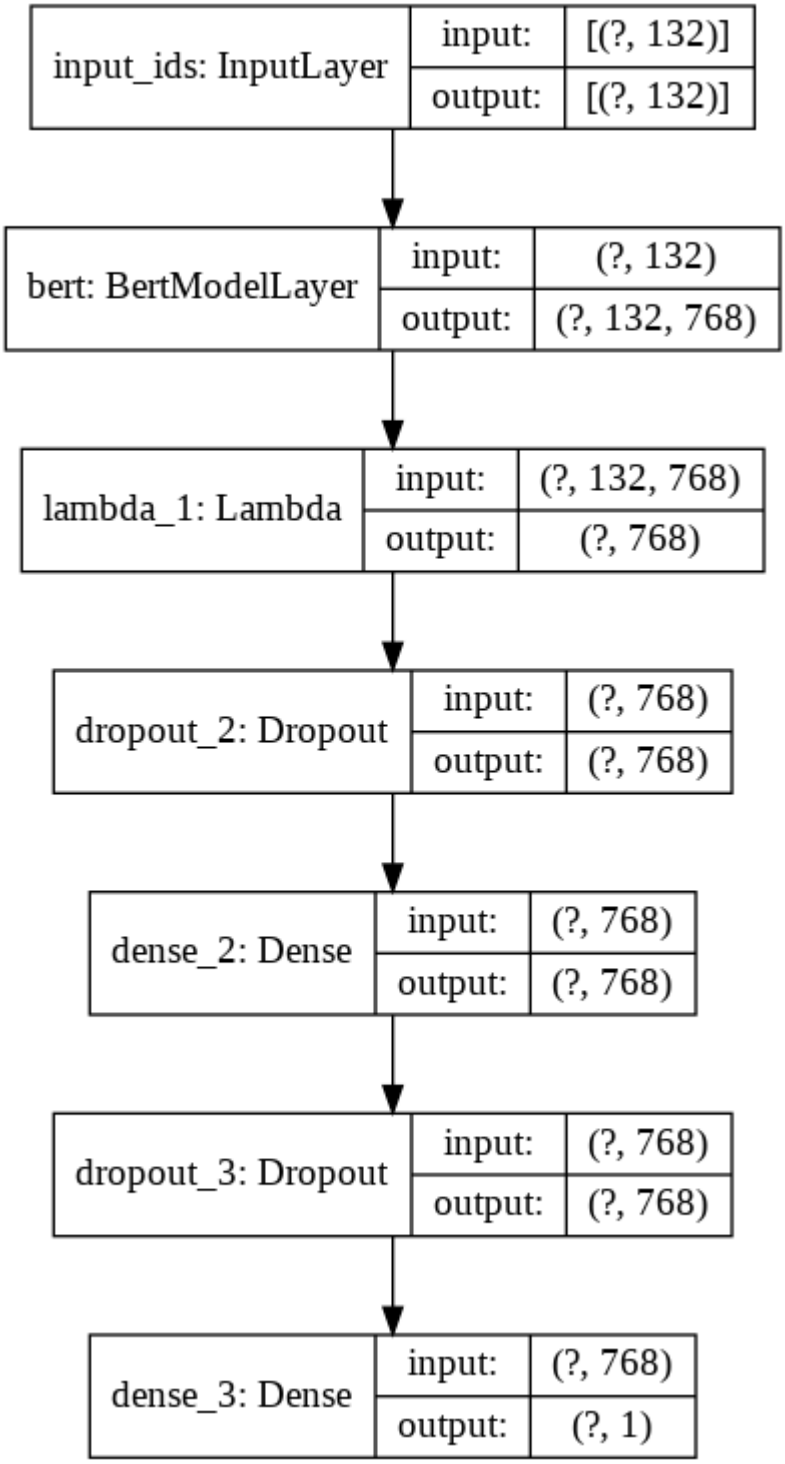
بسمه تعالی

در تحویل پیشین ، تحلیل احساسات را با استفاده از روش word2vec انجام دادیم. این‌بار با استفاده از BERT جملات را دسته‌بندی کردیم.

پس از بارگیری مدل پایه BERT، کلاسی توسعه دادم(TweetSentimentAnalysisData) که با دریافت دادگان test و train، بیشینه طول موردنظر برای بردارها و tokenizer ،توییت‌های موجود در دادگان را tokenize کرده و همه آن‌ها را به بردارهایی با اندازه «« بیشینه طول موردنظر برای بردارها »» تبدیل کند.

سپس با استفاده از tensorflow.keras شبکه عصبی با معماری زیر را طراحی کردم.



شبکه عصبی را با BinaryCrossentropy و Adam به ترتیب به عنوان loss_function و optimizer ایجاد کردم.
سپس مدل را با دادگان پردازش شده که نتیجه فعالیت کلاس TweeterSentimentAnalysisData هستند به صورت زیر آموزش دادم.

```
model.fit(
    x=data.x_train ,
    y=data.y_train ,
    validation_split=0.1 ,
    batch_size = 16 ,
    shuffle = True ,
    epochs = 5 ,
    class_weight = class_weights
)
```

data نمونه‌ای(instance ) از کلاس TweeterSentimentAnalysisData ، و class_weight وزن هر کلاس است.
x_train و y_train به ترتیب از بعد ۱۳۲ * ۲۵۵۶۹ و ۱ *۲۵۵۶۹ می‌باشند.
وزن کلاس منفی (بدون محتوای نژادپرستانه) ۰.۵۳۷۷ و وزن کلاس مثبت (حاوی محتوای نژادپرستانه یا تبعیض جنسیتی) ۷.۱۲۸۰ است.

تصویر زیر نشان‌دهنده خروجی classification_report مدل است.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.97 | 0.98 | 5950 |
| 1 | 0.64 | 0.82 | 0.72 | 444 |
| accuracy | | | 0.96 | 6394 |
| macro avg | 0.81 | 0.89 | 0.85 | 6394 |
| weighted avg | 0.96 | 0.96 | 0.96 | 6394 |

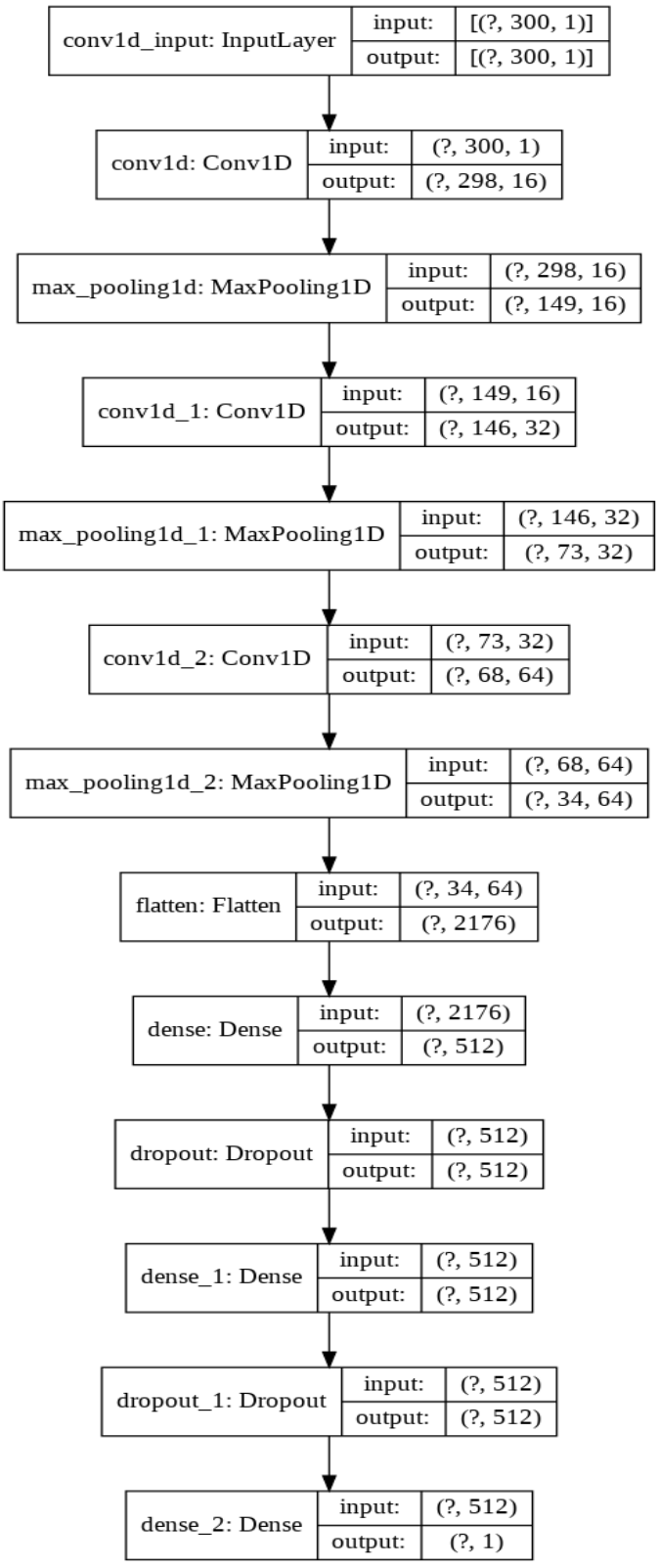
توییت های حاوی اظهارات نژادپرستانه یا تبعیض جنسیتی متعلق به کلاس 1 و دیگر توییت‌ها متعلق به کلاس 0 هستند.
شرح زمان‌های مصرفی برای این classifier به صورت زیر است :

- preprocessing : 18 seconds
- training : 50 minutes
- predicting : 69 seconds

اما برای مقایسه دقیق‌تر بین word2vec و BERT لازم بود مدل روش word2vec را هم به شبکه‌عصبی تغییر دهم زیرا در تحویل قبل از مدل LinearSVC(c=5) برای word2vec استفاده‌کرده‌بودم.

در ادامه به توضیح مدل طراحی شده بر اساس word2vec می‌پردازم .

در گام پیش‌پردازش، شناسه کاربران و علامت # را از جملات حذف کردم. در ادامه باتوجه به POS کلمات را lemmatize ، سپس شبکه‌عصبی با معماری زیر طراحی کرده و با binarycrossentropy به عنوان loss_function و adam به عنوان optimizer آن‌را کامپایل کردم.



مدل را با پارامترهای مشخص به شکل روبرو آموزش دادم :
class_weights و ابعاد x_train و y_train همانند مدل BERT هستند.
تعداد epochs عدد ۱۰ می‌باشد.

```
text_model.fit(
    x_train,
    y_train ,
    validation_split = 0.1 ,
    shuffle = True ,
    batch_size = 16,
    epochs=NB_EPOCHS,
    class_weight = class_weights
)
```

خروجی f1-score و classification-report مدل را در تصویر زیر مشاهده می‌کنید :

```
f1-score equals to  0.5932203389830509

              precision    recall  f1-score   support

           0       0.97      0.98      0.97      5961
           1       0.62      0.57      0.59       432

    accuracy                           0.95      6393
   macro avg       0.80      0.77      0.78      6393
weighted avg       0.95      0.95      0.95      6393
```

مشاهده می‌شود که f1-score در مقایسه با مدل LinearSVC بیش از ۱۰ درصد افزایش داشته است! (f1_score مدل LinearSVC حدودا ۴۸.۹۷ درصد بود )
شرح زمان‌های مصرفی به صورت زیر است :

- preprocessing : 34 seconds
- training : 100 seconds
- predicting : 1 second

مقایسه کلی بین سه روش را در زیر مشاهده می‌کنید :

| | model | preprocess_time(seconds) | training_time(seconds) | predicting_time(seconds) | f1_score(percent) | n_samples_for_train |
|---|---|---|---|---|---|---|
| 0 | word2vec_LinearSVC | 7.65 | 13.28 | 1 | 48.97 | 22373 |
| 1 | word2vec_nn | 34.00 | 100.00 | 1 | 59.33 | 25569 |
| 2 | BERT | 18.00 | 3000.00 | 69 | 72.00 | 25569 |

dataframe این مقایسه در پوشه documentation موجود است.