Sentiment Analysis

FRANCOISE Arnaud

Introduction

Nowadays analyzing social media can be a huge asset for companies.

They can understand the need of clients without asking them drawback. They can express freely they opinion about the company.

In this study we will explore a strategy to study what customers think about a company by predicting whether his comments is positive or negative.

Summary:

I – Data description

A – Dataset content

B- Pre-processing

II – Training a IA model

A- Linear Regression

B- RNN

C- Improvement

I – Description of the Dataset and pre-processing

A – The Dataset

The dataset under study is a csv file containing 1,600,000 tweets extracted using the twitter API. The tweets have been annotated 0 for negative and 4 for positive.

The csv file presents different categories we could study:

1. target: the polarity of the tweet (*0* = negative, *2* = neutral, *4* = positive)
2. ids: The id of the tweet (*2087*)
3. date: the date of the tweet (*Sat May 16 23:58:44 UTC 2009*)
4. flag: The query (*lyx*). If there is no query, then this value is NO_QUERY.
5. user: the user that tweeted (*robotickilldozr*)
6. text: the text of the tweet (*Lyx is cool*)

In order to only predict whether a tweet is positive or negative we will study only the target and the text content.

The idea is to create a label matrix and a “text” matrix.
Of course IA models don’t take “text” as entry. We will have to pre-process all tweet to convert it to vectors

B – Pre-processing

Tokenization:

Transform a text into an array of word as string.

Lemmatization:

Take only the canonical form of a word:

- A verb will be seen as its infinitive form.
- A nom its masculine and singular form.

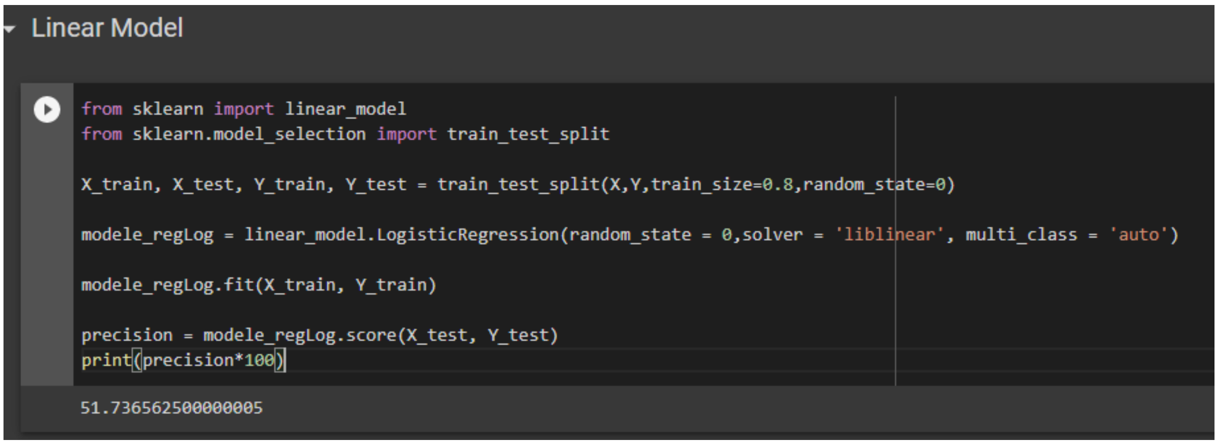
Stopword:

The most frequent word that do not bring more information are considered as noise. We will try to remove it as much as possible.

II – Training an IA model

A- Linear Regression

Figure II.A.1 : Logistic Regression



Linear Model

```
from sklearn import linear_model
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=0.8,random_state=0)

modele_regLog = linear_model.LogisticRegression(random_state = 0,solver = 'liblinear', multi_class = 'auto')

modele_regLog.fit(X_train, Y_train)

precision = modele_regLog.score(X_test, Y_test)
print(precision*100)
```

```
51.736562500000005
```

On this baseline we have about 51.7% accuracy in predicting whether a comment is positive or negative. Let’s see if we can improve this accuracy by developing a Recurrent Neural Network.

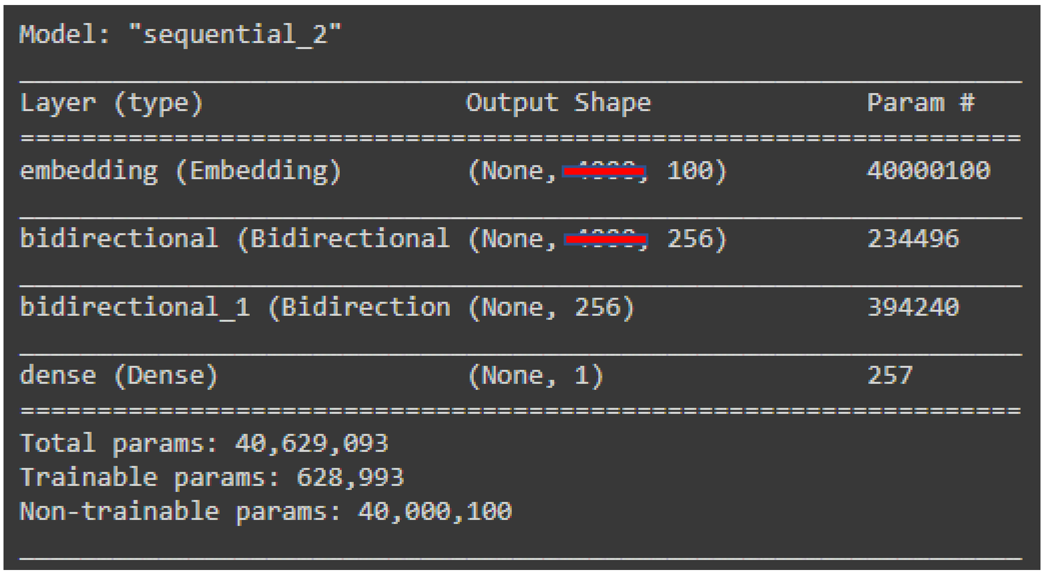
B- RNN

Secondly, we tried to make a Recurrent Neural Network to classify positive and negative tweet.

Here is a basic structure of our model (cf. Figure II.B.1) trained over 80 000 tweets. (dataset/4) .

Noticed that the embedding layer is based on the notion of transfer learning: we use a pretrained layer using GloVe representation.

Figure II.B.1 : Model structure



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, [redacted] 100)         40000100
_________________________________________________________________
bidirectional (Bidirectional (None, [redacted] 256)         234496
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               394240
_________________________________________________________________
dense (Dense)                (None, 1)                 257
=================================================================
Total params: 40,629,093
Trainable params: 628,993
Non-trainable params: 40,000,100
_________________________________________________________________
```

The first model shows this result:

Figure II.B.2 : Model 1 Accuracy

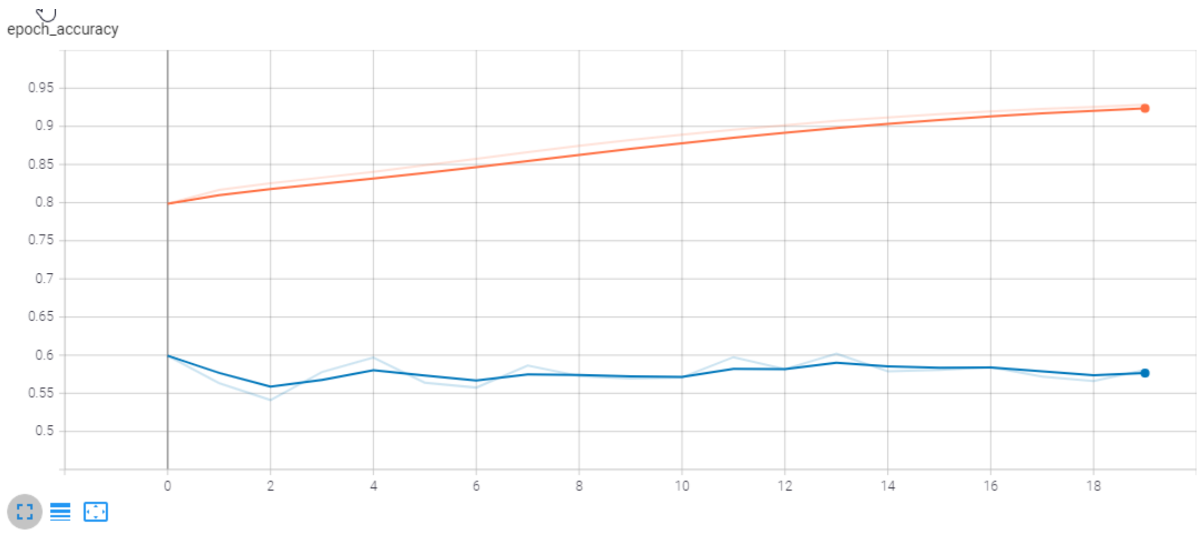
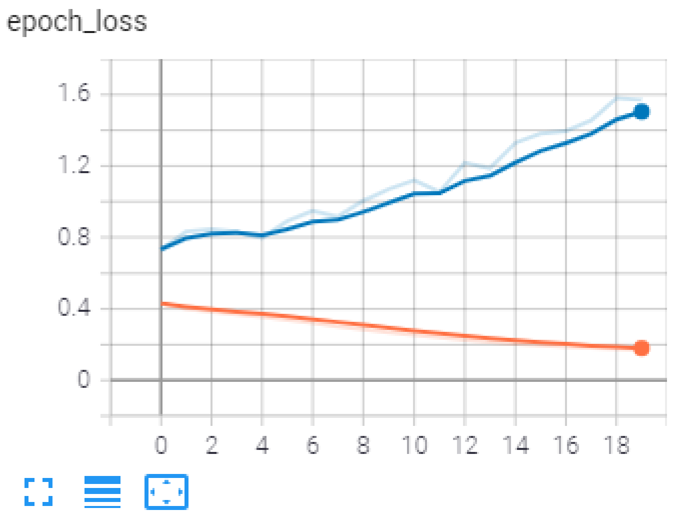


Figure II.B.3 : Model 1 Loss



On figure Figure II.B.2 in orange we have the accuracy during training. This indicates that our algorithm is learning something.

However, the blue curve is the validation accuracy (we evaluate prediction of our model).

We can see that our prediction is about 60% accurate. We will try to improve this rate.

On Figure II.B.3 we can see in orange that the training loss is decreasing then our algorithm is learning from our data.

However, the validation loss is increasing which means that our model seems to overfit and that even if we increase the number of epochs, this won't improve our accuracy.

To improve our model, we should have a decreasing validation loss. When the validation loss will be at its minimum value, we should have this best accuracy rate for this model.

C- Improvement

First thing first we will train different model to see which one is the more adequate for our problem with different parameters.

Here we can investigate how the size of the LSTM layers affect our training.

We will then try different size of LSTM layers: 64, 128, 255 and 1024.

We will also add some dropout to prevent as much as possible overfitting.

Figure II.C.1: Model Optimisation accuracy

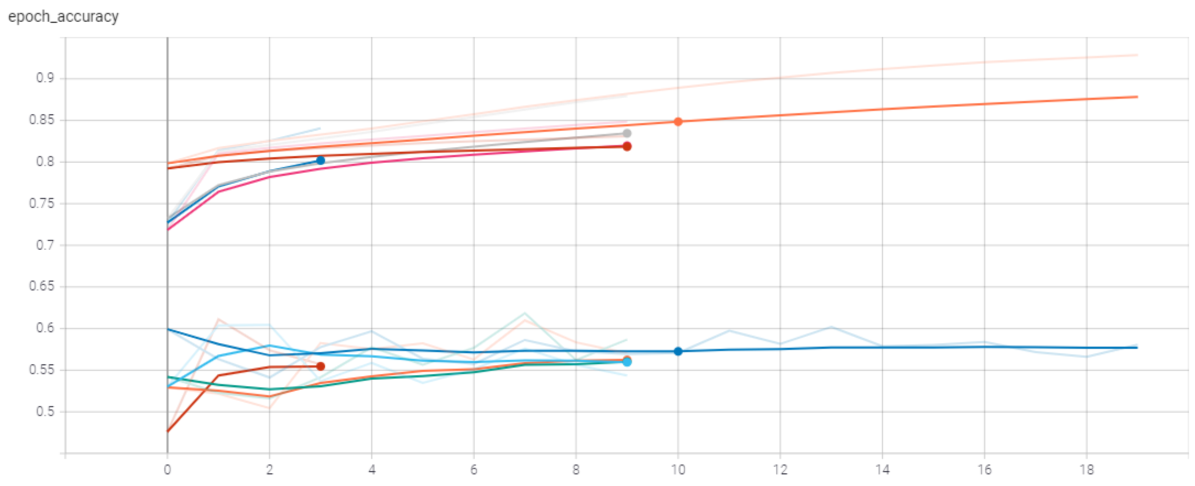
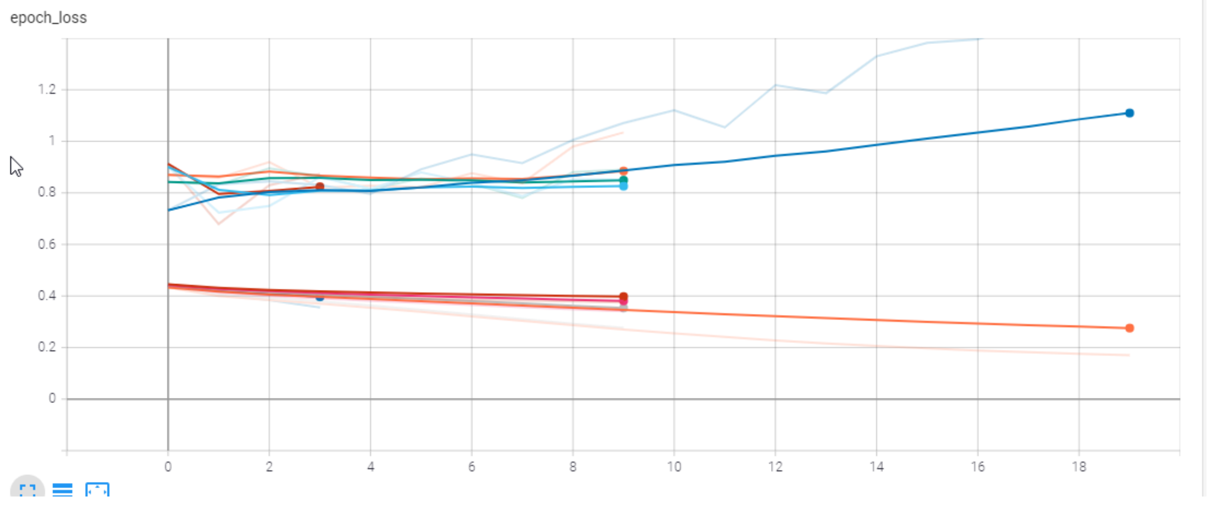


Figure II.C.2: Model Optimisation loss



Our model struggle to get higher than 60% of accuracy. It seems that the size 64 and 1024 seems to have the best behaviour in the epoch loss. We can try to improve the size of the dataset to take into account.

Let's train those two models on longer epochs to compare the two models(cf. STEP2.ipynb):

Figure II.C.3: Model Optimisation2 accuracy

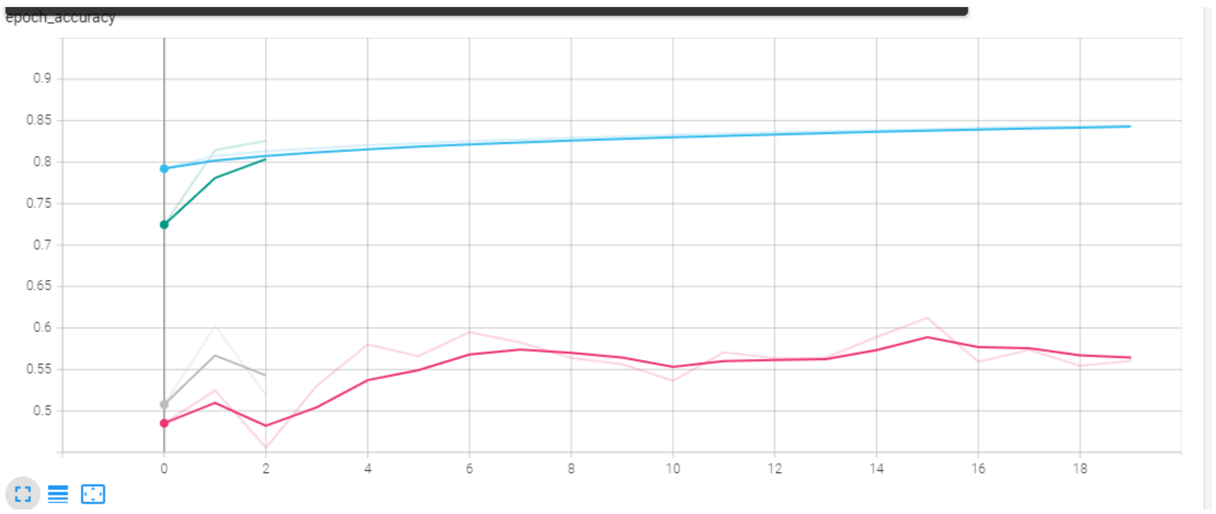
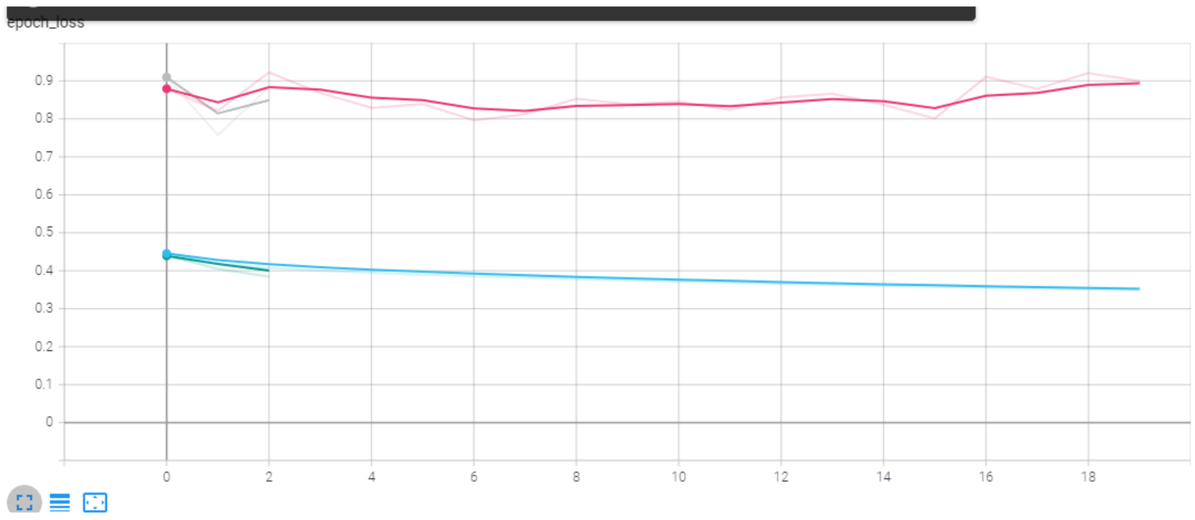


Figure II.C.4: Model Optimisation2 loss



Unfortunately, the model with a LTSM of size 1024 takes 40min per epoch, I stopped it on the third epochs.

However, we may already draw some conclusion.

The model composed of LSTM layers of size 64 are faster to train and the epoch validation loss (cf. Figure II.C.4) seems to globally decrease.

Then we may surmise that by increasing the number of epochs for the Model_64_LSTM we may increase its accuracy.

For now, we have reached a maximum of about 58% accuracy.

We may also consider improving our pre-processing part.

Conclusion

Through this experimentation we developed a strategy to improve model. The Optimisation part can be used in all IA study (CNN, RNN, etc…).

However, we may say that the post processing play a major role in the accuracy of a model when it comes to natural language processing.