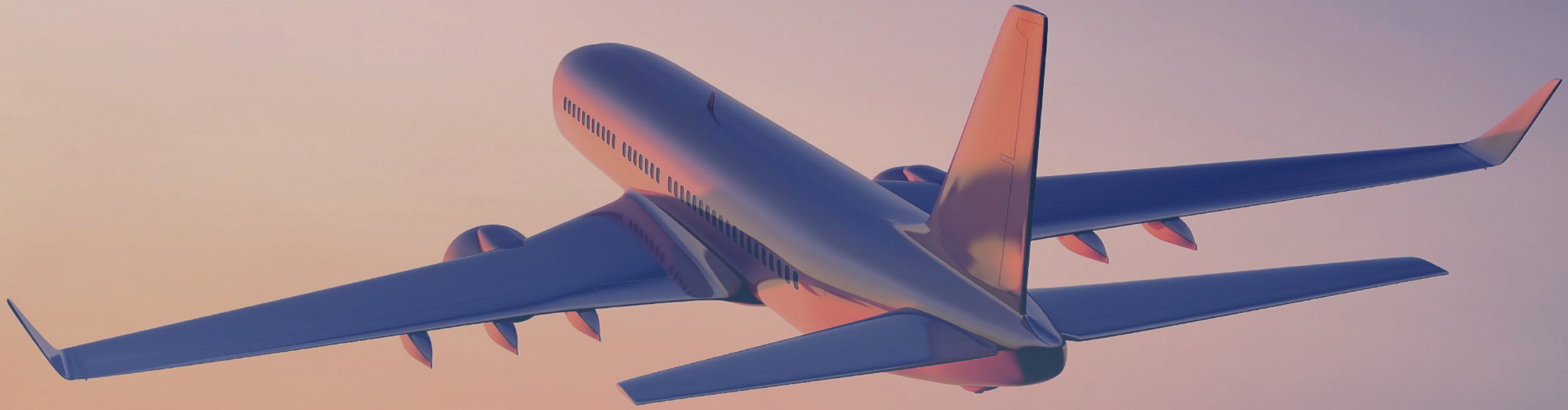
Twitter US Airline Sentiment

Analyze how travelers in February 2015 expressed their feelings on Twitter



Presented by

Dagre Adriani

Gialama Niovi

Kondyli Afroditi

MSc Business Analytics, AUEB

2019

Outline

- Motivation
- Problem & Mission
- Data
- Explanatory Analysis
- Cleaning Phase Methodology
- Models
  - RNNs
  - CNNs
- Conclusions

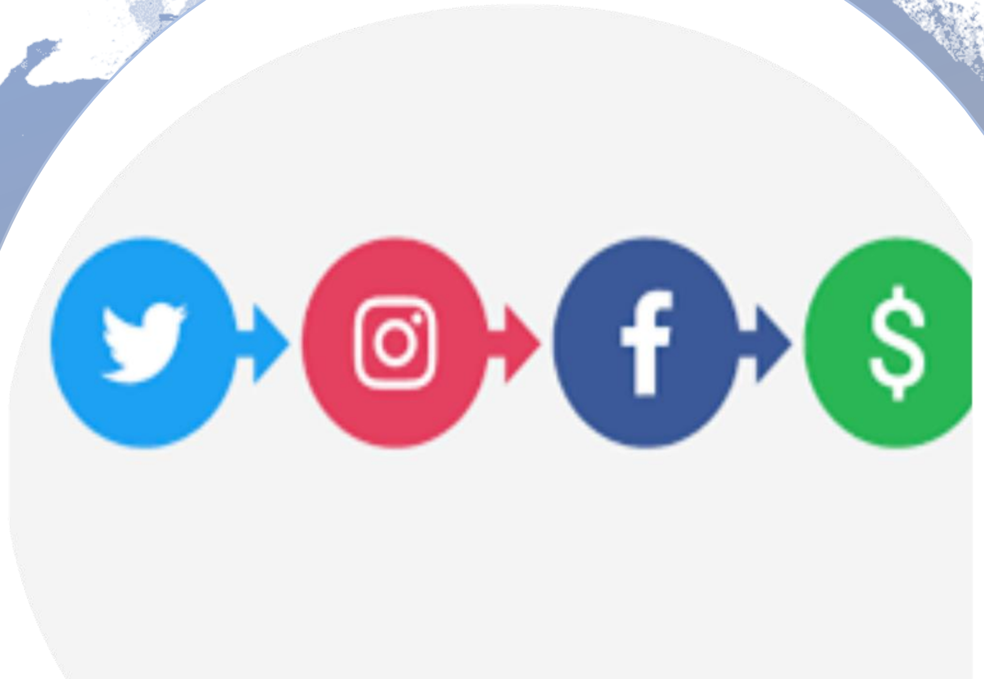
Motivation

- Companies started paying close attention to the voice of customers in order to enhance the customer experience
- Collecting and analyzing customer’s feedback and comments coming from social media about companies themselves, services and products, provides them the advantage to have better information in order to make strategic decisions, while having an accurate understanding of what the customer actually wants and, as a result, a better experience for everyone.
- For this reason, more and more companies deploy sentiment analysis in social media platforms in order to understand what customers like or dislike about the products/services they offer.

Problem & Mission

Companies have to be ready to handle streams of data coming from social media

- The Problem : The vast amount of messages they receive or referred through all the social media platforms
- Our Mission: Build a model that receives Twitter comments and predicts the tweet's writer's sentiment: positive, neutral or negative about the company and/or the providing services
- Who are we: BI services providers appointed by United Airlines in order to an accurate classifier model for all tweets that refer to the company and/or its main rivals.
- What's the Benefit for the company:
  - ✓ extract the appropriate information, so that in the future can predict and prevent any crisis in the Airline sector,
  - ✓ design and accomplish to-the-point strategic moves,
  - ✓ improve customer 's experience and gain better knowledge about their competitors in the Airline sector.

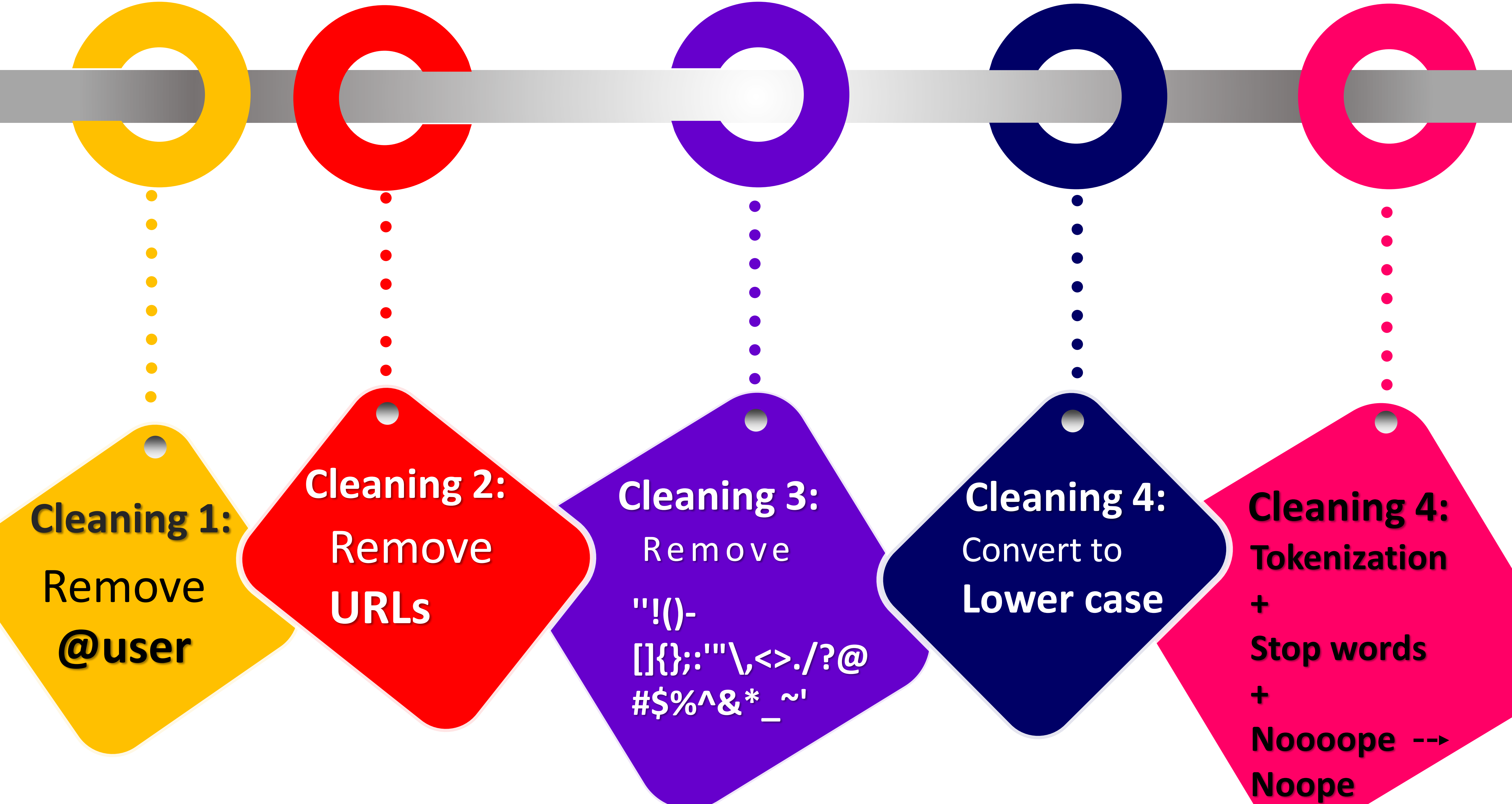


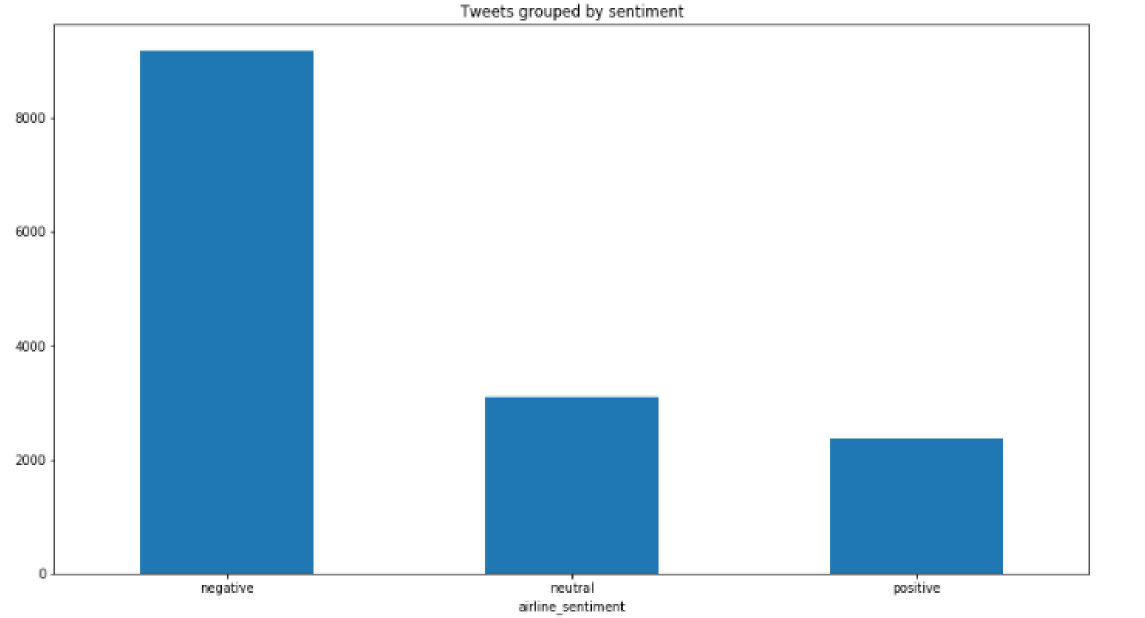
Data

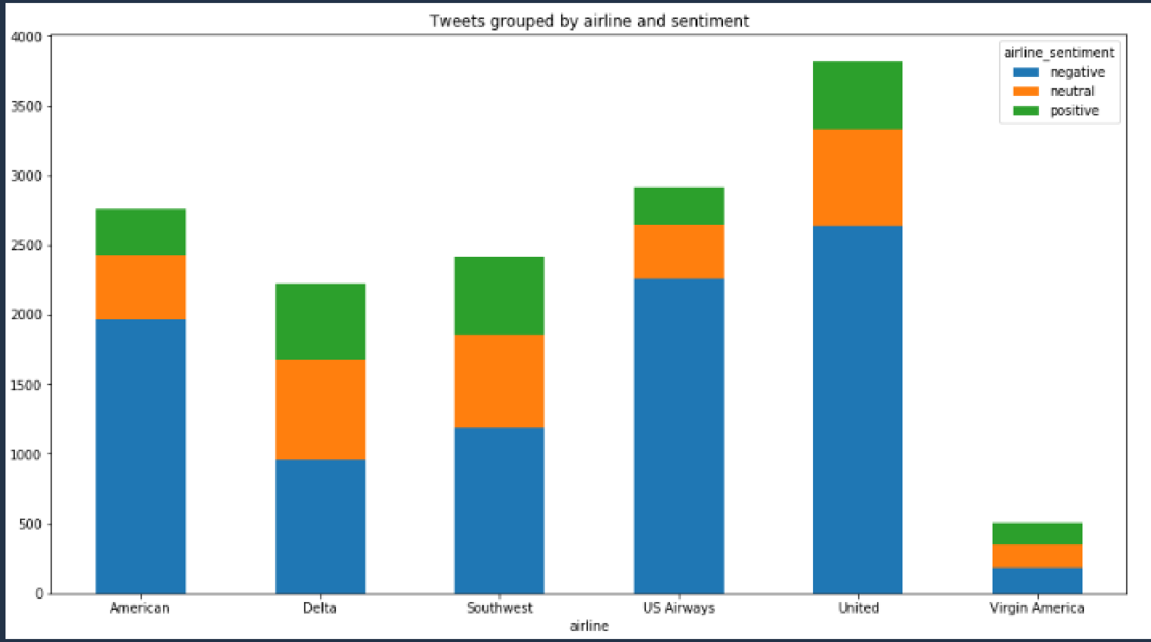
- Tweets were scraped from Twitter in February 2015 about each major US Airline.
- Dataset: "[Twitter US Airline Sentiment]" (from Kaggle as .csv)
- 14,640 rows and 15 columns including:

1. tweet id,
2. sentiment,
3. sentiment confidence score,
4. negative reason,
5. negative reason confidence,
6. airline,
7. sentiment gold,
8. name,
9. retweet count,
10. tweet text,
11. tweet coordinates,
12. time of tweet,
13. date of tweet,
14. tweet location,
15. user time zone.

- Preprocessing was needed in order to gain the best results.
- The stages of this preprocessing were the following:


Cleaning 1:
Remove
@user
Cleaning 2:
Remove
URLs
Cleaning 3:
Remove
"!()-
[]{};:'"\,<>./?@
#$%^&*_~'
Cleaning 4:
Convert to
Lower case
Cleaning 4:
Tokenization
+
Stop words
+
Nooooope --> Noope





Exploratory Analysis

Common reasons for negative comments

	negativereason	airline
3	Customer Service Issue	2910
7	Late Flight	1665
1	Can't Tell	1190
2	Cancelled Flight	847
8	Lost Luggage	724
0	Bad Flight	580
6	Flight Booking Problems	529
5	Flight Attendant Complaints	481
9	longlines	178
4	Damaged Luggage	74

```
[[   0    0    0 ...    0    0 122]
 [   0    0    0 ...  400  928 104]
 [   0    0    0 ...   62   70  96]
 ...
 [   0    0    0 ...  491  323  23]
 [   0    0    0 ...   30  887  42]
 [   0    0    0 ...   66   93   1]]
```

The form of our data that fed the model

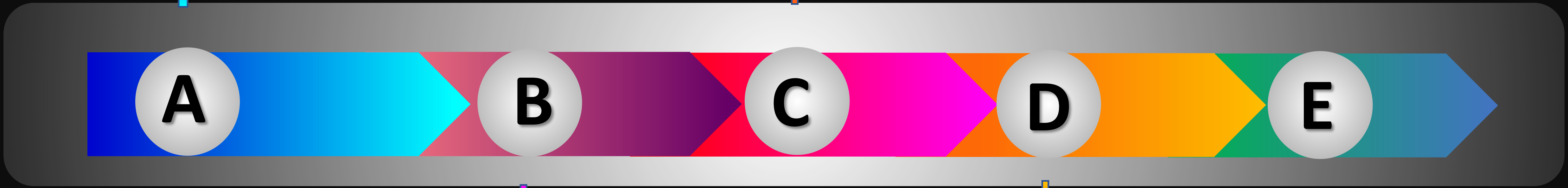
Convolution Neural Networks (CNNs)

Model Inputs

MAX_WORDS = 6000
TOKENIZETION :
Finally, each words is represented by integers based on vocabulary

Compile

Plots



ZERO PADDING
SPLIT DATASET :
- **TRAIN :80% (11712,40)**
- **TEST : 20% (2928,40)**

**** VALIDATION: use the test data**

Build CNN Model

Train and Predictions

B

Build the CNN Model

Embedding Layer

From scratch

Manage lower dimension

INPUT : 45 X 6000*

**Top most common words to consider*

OUTPUT : 45 X 32

Convolution Layer

1DConv -> strides Vertically

Apply 64 filters 2 x 32

Output 40 x 64

Max Pooling

Take the Max Value from each filter pixel

Output 64 x 1

Dropout

Avoid overfitting

20 % in our case

Dense Layer

Give a probability to each sentiment

3 Neurons

Softmax Function

C

Compiling The Model

- **OPTIMIZER**

  Optimal Weigths

- **LOSS FUNCTION**

  Deviation from actual Y values

- **METRICS**

✓ Adam ()

✓ Multiclass Problem ⇒ Categorical_crossentropy

✓ Accuracy ⇒ Better interpretation

D

Train CNN Model + Make Predictions

X_Train, encoded_Y
X_test, encoded_Y_test

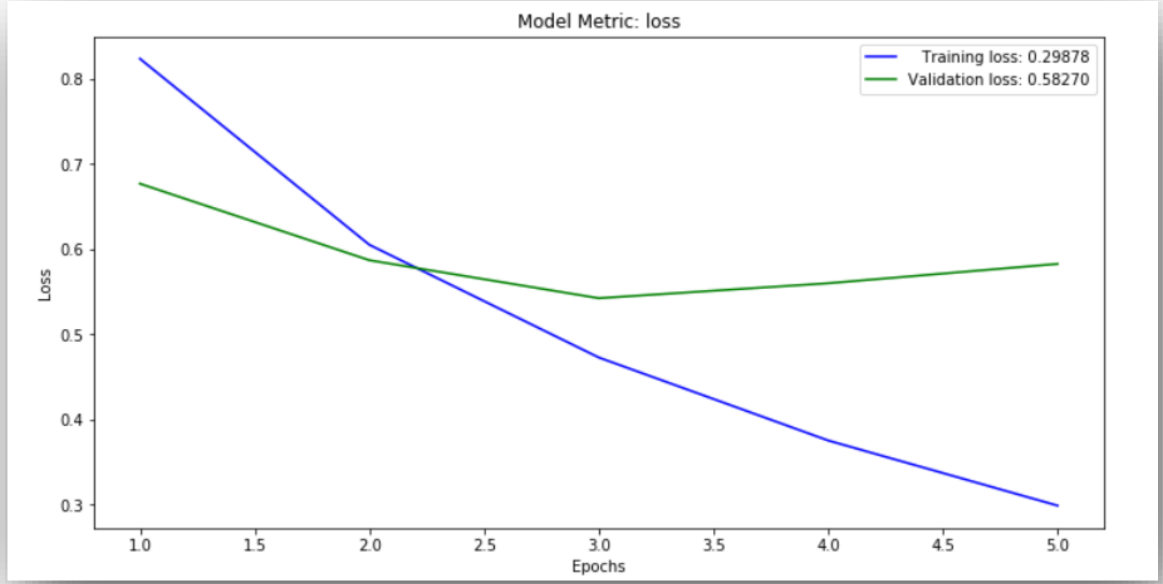
5 epochs
Batch size = 100 tweets

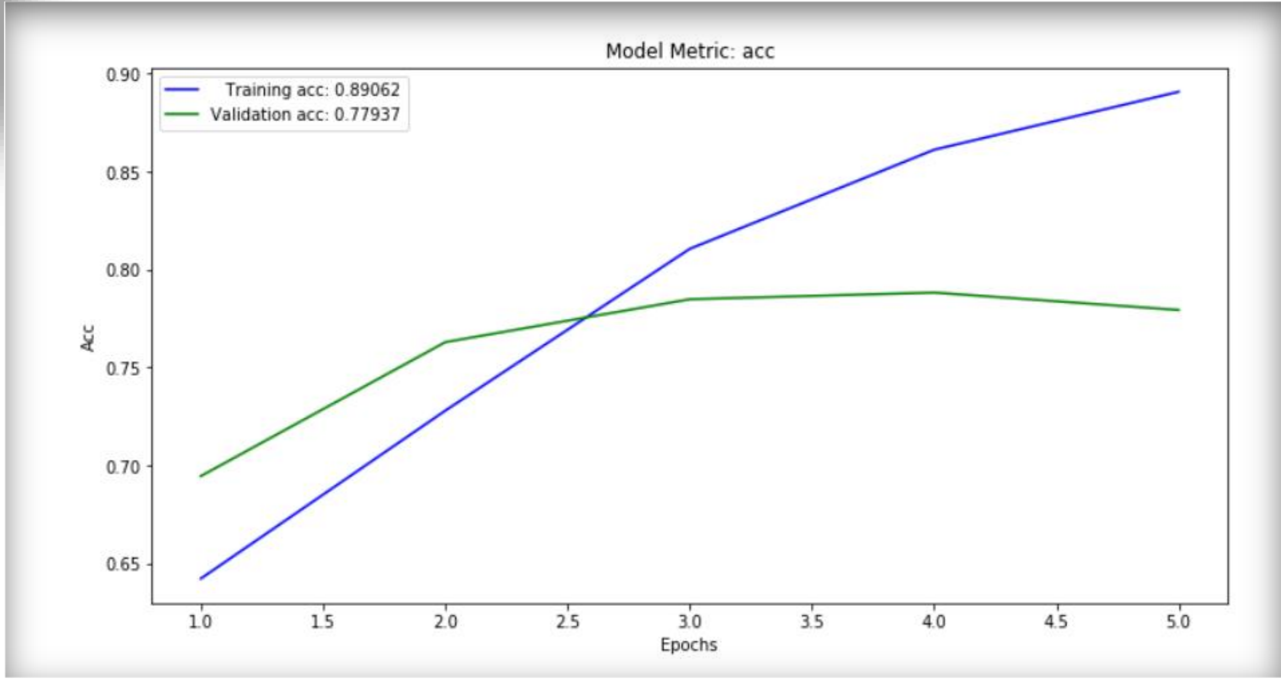
- The best validation accuracy: 79.23 % &
- The minimum loss function: 52.88%

X_Test, Y_Test
Accuracy : 77.9%

E

Plots





Eliminate epochs to 3 and Make a New model :

- ✓ 128 filters 3 x 32
- ✓ Smaller accuracy

RNN Model – Data preprocessing

01

We used regular expressions to remove certain words.

Example:

The airline name at the beginning of each tweet, any URL in the tweet body, symbols and single letters or numbers.

02

We converted all letters to lowercase.

03

We removed all common words that do not have any significant semantic meaning.

Example:

“of”, “over”, “than” etc.

04

We conducted lemmatization, which is the transformation of each word into a lemma.

Example:

“Playint” -> Play

“Plays” -> Play

”Played” -> Play

“Am”, “are”, “is” -> be

“Car”, “cars”, “car’s", "cars’" -> car

RNN Model – Model description

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_36 (Embedding)     (None, 21, 96)            288000
_________________________________________________________________
lstm_36 (LSTM)               (None, 96)                74112
_________________________________________________________________
dense_36 (Dense)             (None, 3)                 291
=================================================================
Total params: 362,403
Trainable params: 362,403
Non-trainable params: 0
_________________________________________________________________
None
```

01

Embedding Layer

A transformation layer used to turn the indexes we provide into dense vectors of fixed size. The embedding space we chose had 96 dimensions, because, this value gave us the better results by using a trial and error technique.

02

LSTM (Long Short-Term Memory) layer

This is the heart of our RNN model. It has 96 memory units and some dropout values to avoid overfitting.

03

Dense Layer

A layer used to change the dimensions of our vector from 96 to just 3, as many as our label categories are.

RNN Model – Input, Loss Function, Optimizer

1. Input

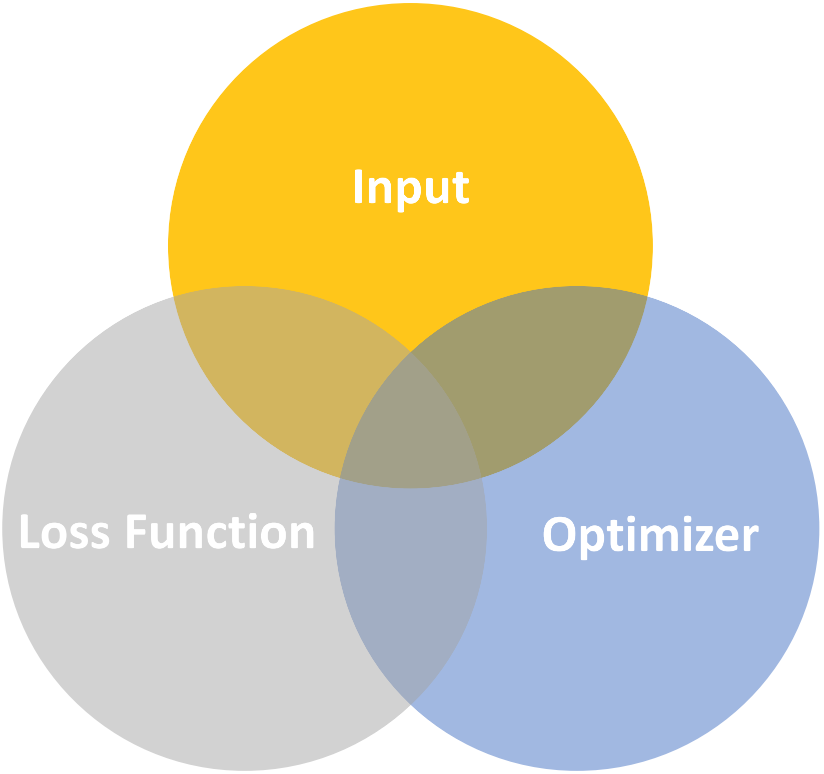
We divided our data into training, validation and testing groups. The ratio we used is 68% : 7% : 25%.

2. Loss Function

We used Multi-Class cross entropy (Keras: categorical crossentropy) function as it fits better multiclass problems such as ours.

3. Optimizer

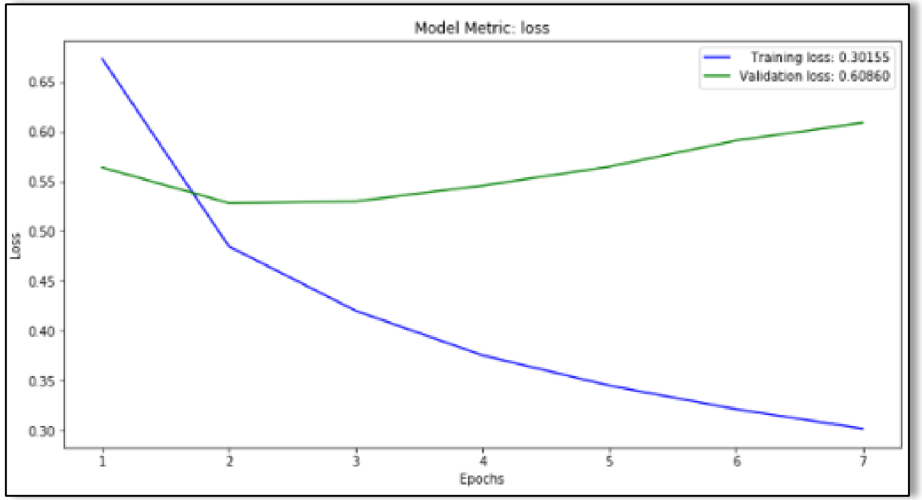
We used Adagrad optimizer by using try and error technique. It gave us the better accuracy on the validation set.

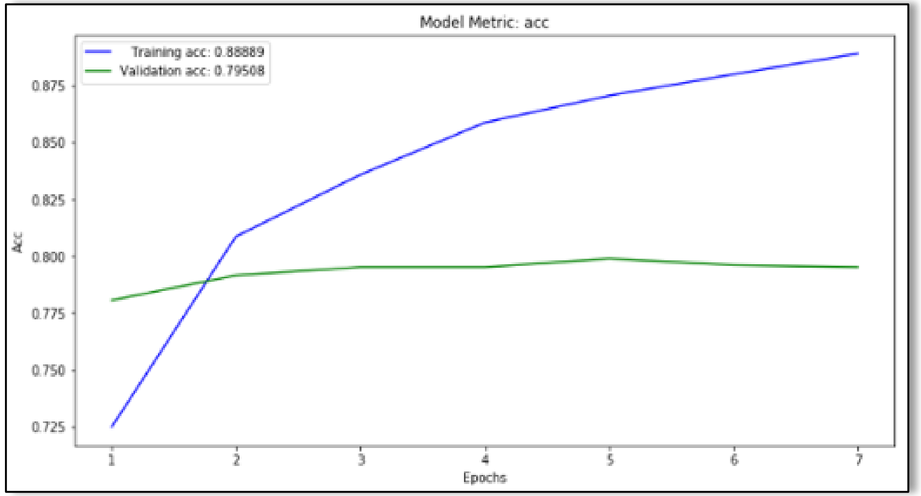


RNN Model - Results

- ✓ 10 epochs
- ✓ trigger when two consecutive validation accuracy values are below the maximum value
- ✓ maximum value we get on the validation set is 79.87%

```
Train on 9882 samples, validate on 1098 samples
Epoch 1/10
9882/9882 [==============================] - 36s 4ms/step - loss: 0.6728 - acc: 0.7249 - val_loss: 0.5638 - val_acc: 0.7805
Epoch 2/10
9882/9882 [==============================] - 23s 2ms/step - loss: 0.4845 - acc: 0.8085 - val_loss: 0.5280 - val_acc: 0.7914
Epoch 3/10
9882/9882 [==============================] - 24s 2ms/step - loss: 0.4199 - acc: 0.8357 - val_loss: 0.5298 - val_acc: 0.7951
Epoch 4/10
9882/9882 [==============================] - 23s 2ms/step - loss: 0.3752 - acc: 0.8585 - val_loss: 0.5455 - val_acc: 0.7951
Epoch 5/10
9882/9882 [==============================] - 23s 2ms/step - loss: 0.3448 - acc: 0.8704 - val_loss: 0.5646 - val_acc: 0.7987
Epoch 6/10
9882/9882 [==============================] - 23s 2ms/step - loss: 0.3211 - acc: 0.8798 - val_loss: 0.5907 - val_acc: 0.7960
Epoch 7/10
9882/9882 [==============================] - 23s 2ms/step - loss: 0.3015 - acc: 0.8889 - val_loss: 0.6086 - val_acc: 0.7951
```





Conclusions

Accuracy CNN vs RNN : 77.9% vs 77.5 %

CNN filters find a specific pattern in a tweet

BUT

Fail if there are not words connected with sentiment analysis

SO

Keep **RNN** to capture the important information for the right classification


THANK
YOU!