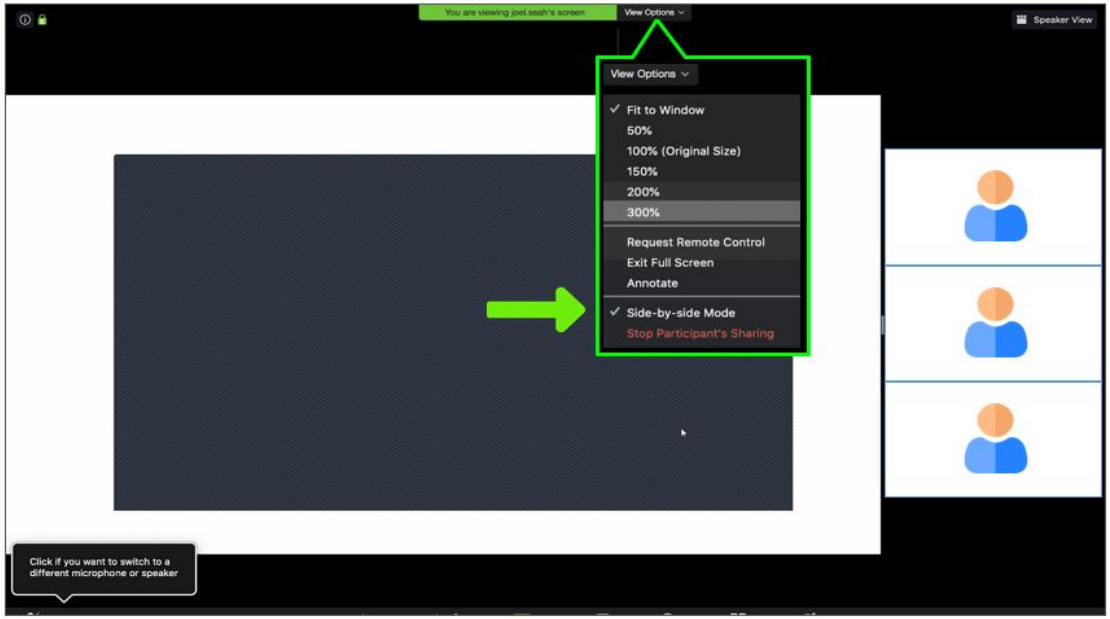
We'll Be Starting Shortly!

To help us run the workshop smoothly, kindly:

- Submit all questions using the Q&A function
- If you have an urgent request, please use the "Raise Hand" function

Using Zoom: Viewing Mode



Side-By-Side Mode

- When sharing screen (slide share)
- With small thumbnails of people on the sidebar

STEPS:

1. View Options
2. Side-By-Side Mode

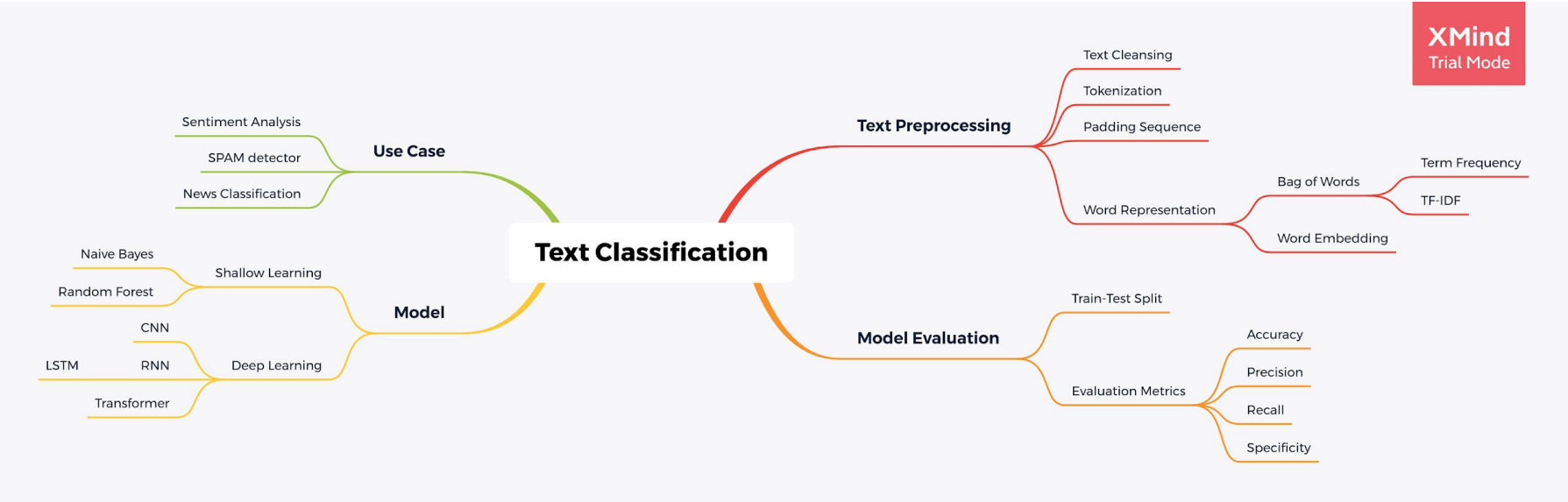
Text Classification with Deep Learning

Concept and Use Case

Visit Our Website

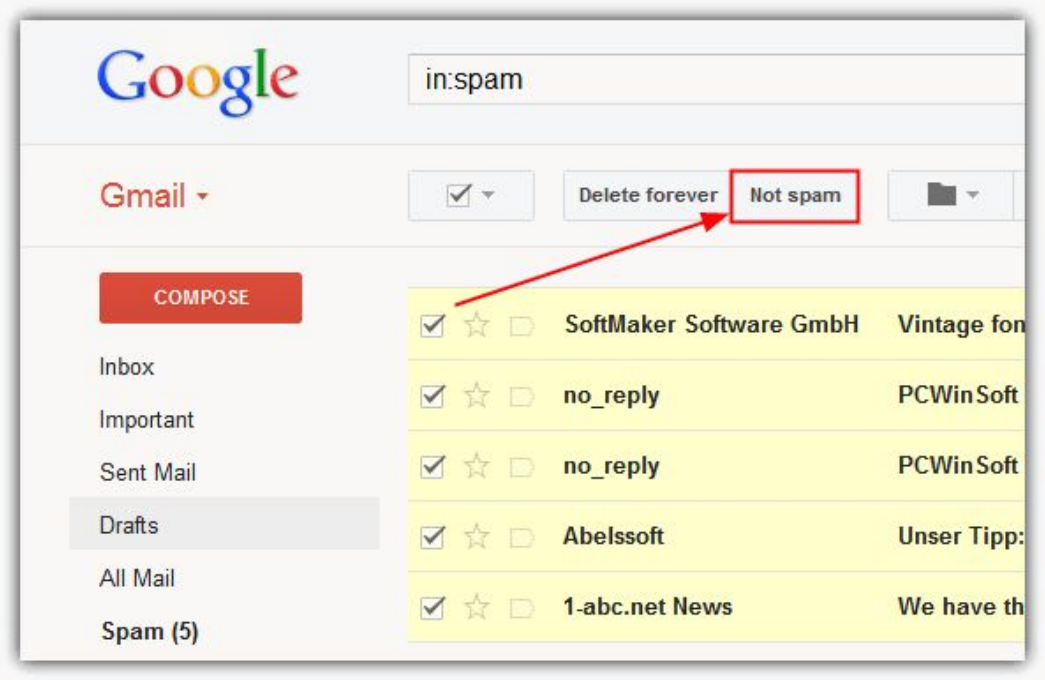
Visit Our Blog

Workshop Objective

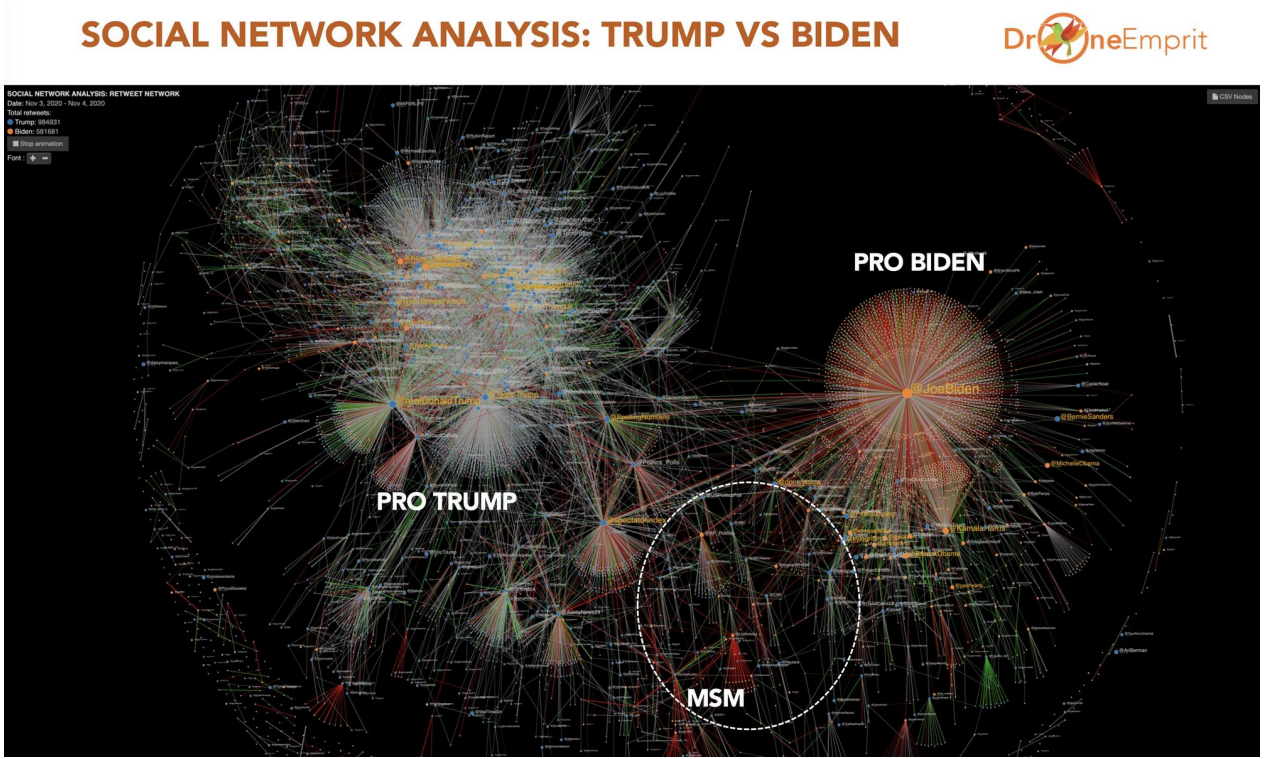


Why Classify Text?

SPAM classification



Why Classify Text?



Why Classify Text?

Characterizing public emotions and sentiments in COVID-19 environment: A case study of India

Figure 4. Wordcloud of top 20 topics from positive tweet group.

Figure 5. Wordcloud of top 20 topics from negative tweet group.

Development of Text Classification Model

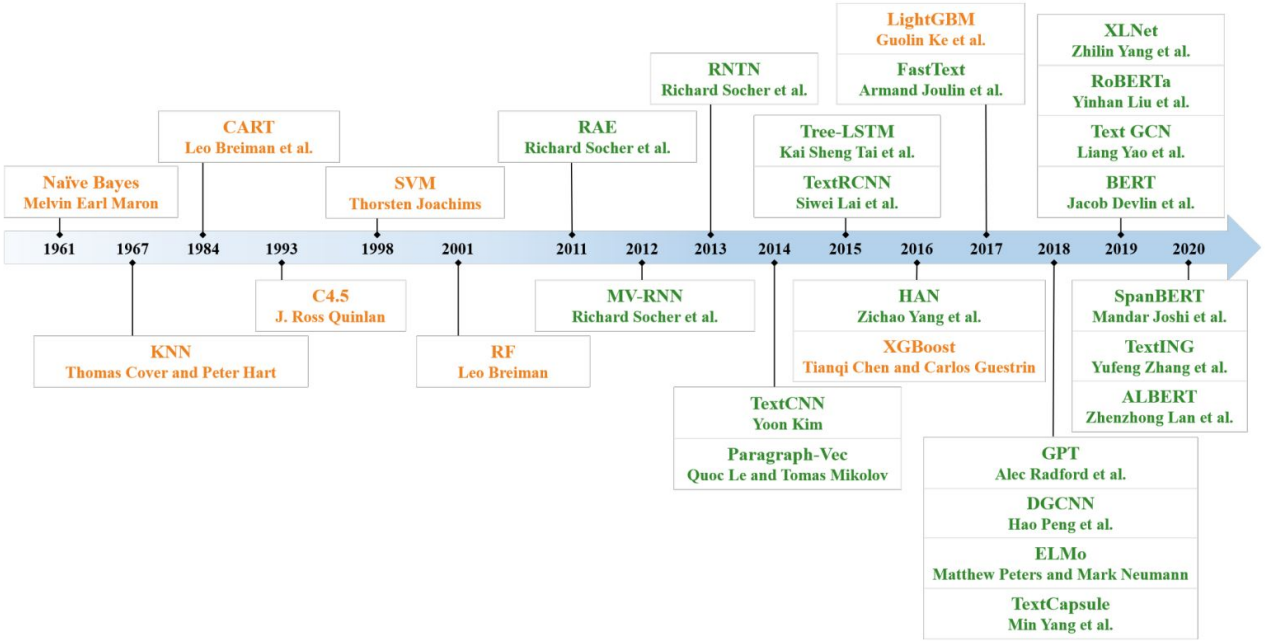


Fig. 2. Schematic illustration of the primary text classification methods from 1961 to 2020. Before 2010, almost all existing methods are based on shallow models (orange color); since 2010, most work in this area has concentrated on deep learning schemes (green color).

Development of Text Classification Model

Text Classification

Edit Task

Natural Language Processing

401 papers with code **29** benchmarks

About

Edit

Text classification is the task of assigning a sentence or document an appropriate category. The categories depend on the chosen dataset and can range from topics.

(Image credit: Text Classification Algorithms: A Survey)

Benchmarks

Add a Result

TREND	DATASET	BEST METHOD	PAPER TITLE	PAPER	CODE	COMPARE
	AG News	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding			See all
	DBpedia	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding			See all
	TREC-6	USE_T+CNN	Universal Sentence Encoder			See all
	20NEWS	SSGC	Simple Spectral Graph Convolution			See all
	IMDb	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding			See all

How Machine Classify Text?

Positive?

Best hunting game on the market as of the writing of this review.

Negative?

The bug in this game is unbearable and annoying.

Shallow Learning with Naive Bayes

Bag of Words Method

Step of Text Cleansing

1. Make all text lowercase
2. Remove punctuation
3. Remove stopwords (frequent and unnecessary words)
4. Stemming

Best hunting game on the market as of the writing of this review.

The bug in this game is unbearable and annoying!!!!

best hunting game on the market as of the writing of this review

the bug in this game is unbearable and annoying

best hunting game market write review

bug game unbearable annoy

Shallow Learning with Naive Bayes

Bag of Words Method

best hunting game market write review

bug game unbearable annoying

Tokenization

Split sentence into individual terms/words

best	hunting	game	market	write	review

Document-Term Matrix

Create a matrix between each document and each text

bug	game	unbearable	annoy

Docs	best	hunting	game	market	write	review	bug	unbearable	annoy
1	1	1	1	1	1	1	0	0	0
2	0	0	1	0	0	0	1	1	1

algoritma
data science education center

Shallow Learning with Naive Bayes

Bag of Words Method

Term-Frequency (TF)

Counts the frequency of each terms in each document/sentence.

Docs	best	hunting	game	market	write	review	bug	unbearable	annoy
1	1	1	1	1	1	1	0	0	0
2	0	0	1	0	0	0	1	1	1

Term Frequency / Inverse Document Frequency (TF-IDF)

Measure how important a word is to a document in a collection (or corpus) of documents

$$W(d,t) = TF(d,t) * log(\frac{N}{df(t)})$$

Docs	best	hunting	game	market	write	review	bug	unbearable	annoy
1	0.115	0.115	0	0.115	0.115	0.115	0	0	0
2	0	0	0	0	0	0	0.173	0.173	0.173

Shallow Learning with Naive Bayes

Bag of Words Method

Pros

1. Simple
2. Quick to train the model with Naive Bayes

Cons

1. Ignore information of words as sequence
2. Poor scalability
3. Ignore similarity of words

this is good = is this good

Case Study

Classifying Steam Reviews with Deep Learning

algoritma
data science education center

Library Requirement

R

Data Wrangling

- `tidyverse`

Text Preprocessing

- `tidytext`
- `textclean (require pacman)`
- `hunspell`

Model Fitting

- `keras`
- `e1071`

Model Evaluation

- `yardstick`

python / anaconda

`tensorflow 2.2.0 or`
`tensorflow 2.0.0`

The Dataset

Import the Dataset

```
df <- read.csv("data/steam_review.csv")
```

Inspect the Dataset

```
glimpse(df)
```

```
Rows: 17,494
Columns: 5
$ review_id       <int> id of the review
$ title           <chr> Title of the game
$ year            <int> Year in which the review was posted
$ user_review     <chr> Review of the user
$ user_suggestion <int> Recommended(1) and Not Recommended(0)
```

	review_id	title	year	user_review	user_suggestion
1	10787	Trove	2017	ive played this game for a while now.I'd say this is one of the best free mmo type game ive pla...	1
2	1565	Fractured Space	2016	Do you enjoy true competition in a balanced environment where nobody has an advantage tha...	1
3	25010	Cuisine Royale	2018	Early Access ReviewIdea is good , controls are clunky . You can shoot someone point blank with...	0
4	6513	SMITE®	2015	I know my steam play time is low but I've been playing this game for a solid amount of time be...	1
5	22167	Robocraft	2016	Early Access ReviewRobocraft. Robocraft is meant to be a fun enjoyable game but now this ga...	0
6	2863	Path of Exile	2018	This is the best game ever, i have skipped school like 15 times just to play this game all day, n...	1
7	18906	Fallout Shelter	2017	It seems Bethesda have their hearts set out on making video games that are so unhealthily ad...	1
8	8726	Heroes & Generals	2016	Early Access Reviewive pla'ed this game 2yrs ago and it had potential but its like a plane saili...	0
9	15897	theHunter Classic	2014	THIS IS NOT A "FREE TO PLAY GAME" IT IS A TRIAL TO A PAY TO PLAY/SUBSCRIPTION BASED GA...	1
10	9012	Heroes & Generals	2016	Unplayable unless you pay to win. Your guns will do no damage and you'll be killed from across...	0

Text Cleansing

Create Cleansing Function

```
cleansing_text <- function(x) x %>%
                        replace_non_ascii() %>%
                        tolower() %>%
                        str_replace_all(pattern = "\\@.*? |\\@.*?[:punct:]", replacement = " ") %>%
                        str_remove(pattern = "early access review") %>%
                        replace_url() %>%
                        replace_hash() %>%
                        replace_html() %>%
                        replace_contraction() %>%
                        replace_word_elongation() %>%
                        str_replace_all("\\?", " questionmark") %>%
                        str_replace_all("\\!", " exclamationmark") %>%
                        str_replace_all("[:punct:]", " ") %>%
                        str_replace_all("[:digit:]", " ") %>%
                        str_trim() %>%
                        str_squish()
```

Text cleansing in order

1. Remove all non ASCII characters
2. Make all characters lowercase
3. Remove all mention (@….)
4. Remove word *“early access review”*
5. Remove all https or url link
6. Remove all hashtag (#…)
7. Remove all html tag ( )
8. Replace a contracted word (i’m => i am)
9. Remove any word elongation (aaaa => a)
10. Replace ? into word *“quetionmark”*
11. Replace ! into word *“exclamationmark”*
12. Remove all punctuation
13. Remove all numbers
14. Remove unnecessary white space

Text Cleansing

Set Parallel Computing for Faster Cleansing

```
library(furrr)
plan(multisession, workers = 4) # Using 4 CPU cores
```

Apply Text Cleansing

```
df_clean <- df %>%
  mutate(text_clean = user_review %>%
      future_map_chr(cleansing_text)
        )
```

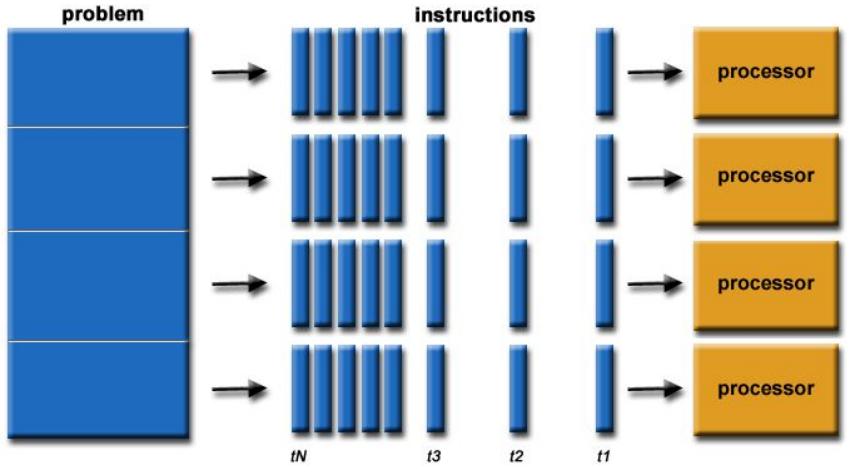
Check Sentence Length

```
word_count <- map_dbl(df_clean$text_clean, function(x) str_split(x, " ") %>%
                        unlist() %>%
                        length()
                      )
```

Filter : Only Use Text with More Than 3 Words

```
df_clean <- df_clean %>%
  filter(word_count > 3)
```

With parallel computing, you can run multiple task simultaneously by using all your CPU cores



Cross-Validation

Set Random Seed

```
set.seed(123)
```

Get Number of Row

```
row_data <- nrow(df_clean)
```

Sample Data with 80% Data as Data Train

```
index <- sample(row_data, row_data*0.8)
data_train <- df_clean[ index, ]
data_test <- df_clean[-index, ]
```

80% Learning

20% Testing

Tokenization

Set Number of Words for Vocabulary

```
Num_words <- 40000
```

Create Text Tokenizer from Data Train Corpus

```
tokenizer <- text_tokenizer(num_words = num_words) %>%
  fit_text_tokenizer(data_train$text_clean)
```

Maximum Length of Words in a Sequence

```
maxlen <- 250
```

Padding and Truncating Text Sequence

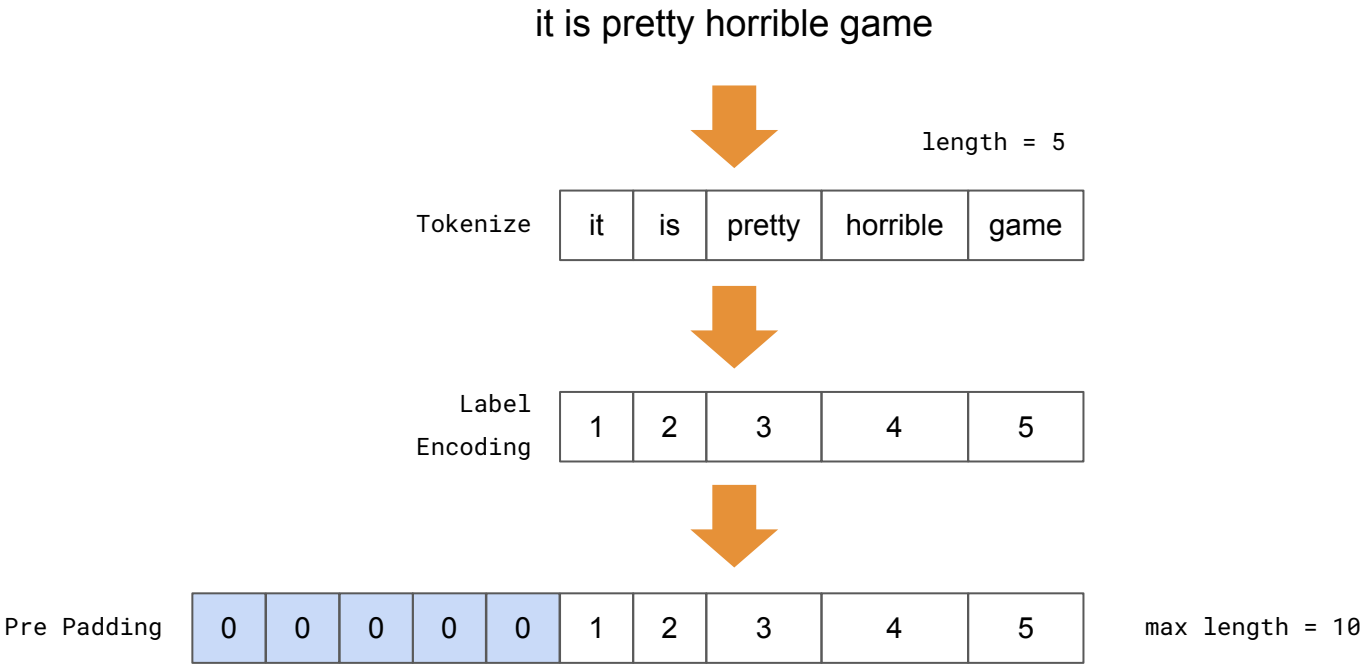
```
train_x <- texts_to_sequences(tokenizer, data_train$text_clean) %>%
  pad_sequences(maxlen = maxlen, padding = "pre", truncating = "post")

test_x <- texts_to_sequences(tokenizer, data_test$text_clean) %>%
  pad_sequences(maxlen = maxlen, padding = "pre", truncating = "post")
```

Take the Target Variable from Data Train

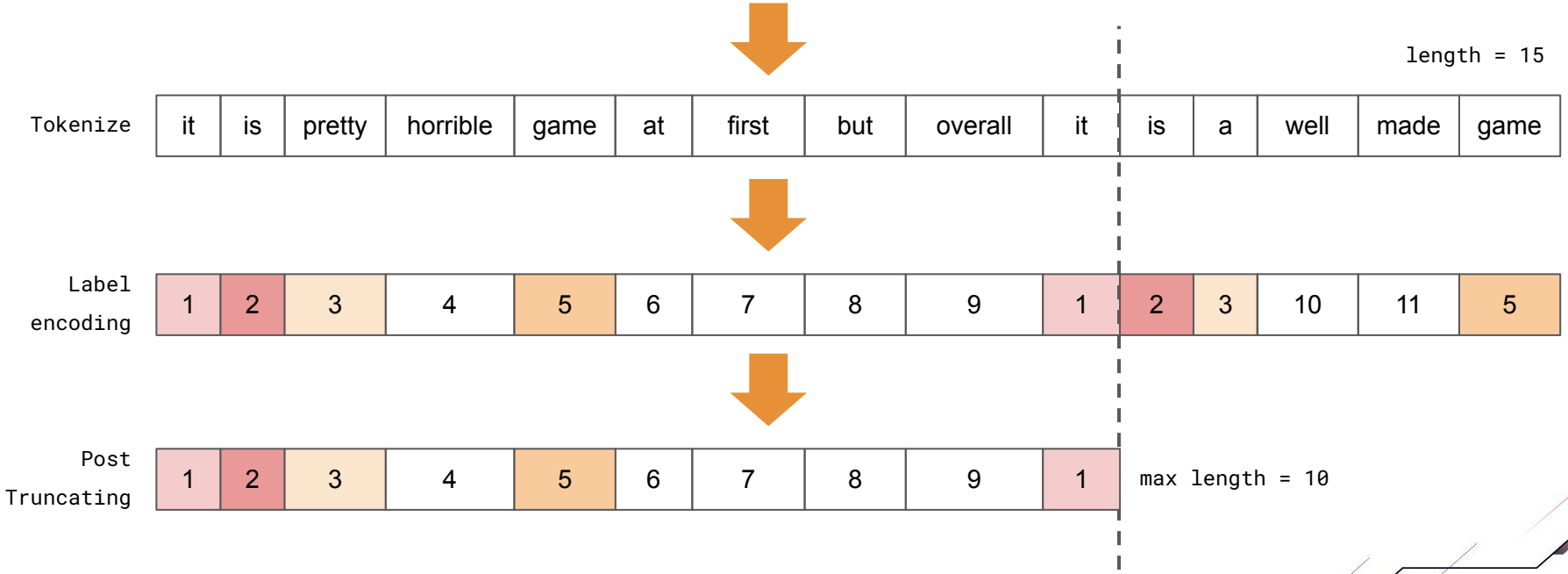
```
train_y <- data_train$user_suggestion
```

Padding and Truncating Sequence



Padding and Truncating Sequence

it is pretty horrible game at first but overall it is a well made game



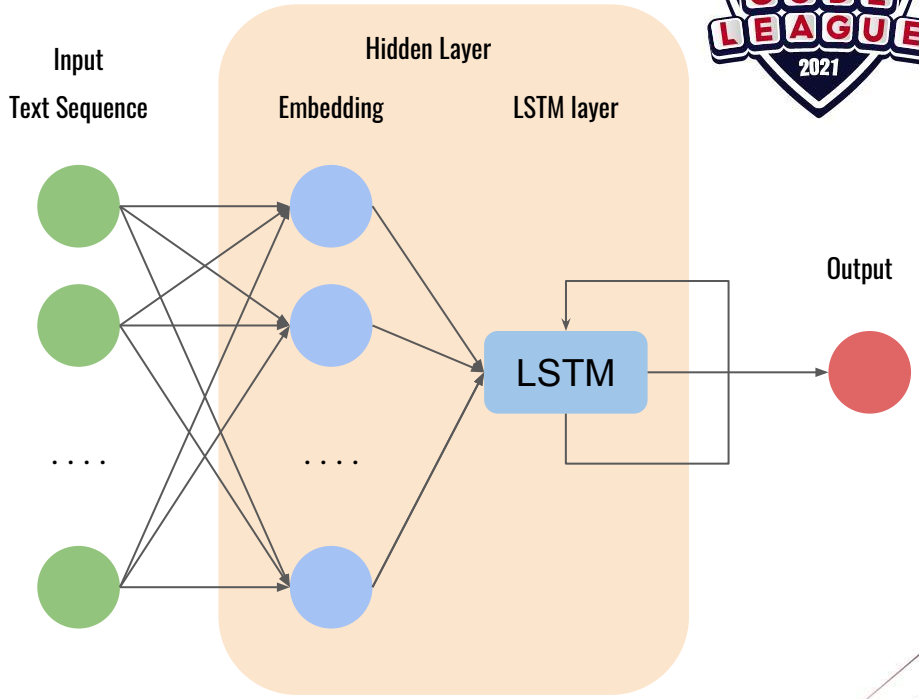
Model Architecture

Set Random Seed for Initial Weight

```
tensorflow::tf$random$set_seed(123)
```

Build model architecture

```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = num_words,
                  input_length = maxlen,
                  output_dim = 8
                  ) %>%
  layer_lstm(units = 8,
             kernel_regularizer = regularizer_l1_l2(l1 = 0.05, l2 = 0.05),
             return_sequences = F
             ) %>%
  layer_dense(units = 1, activation = "sigmoid")
```



Model Architecture

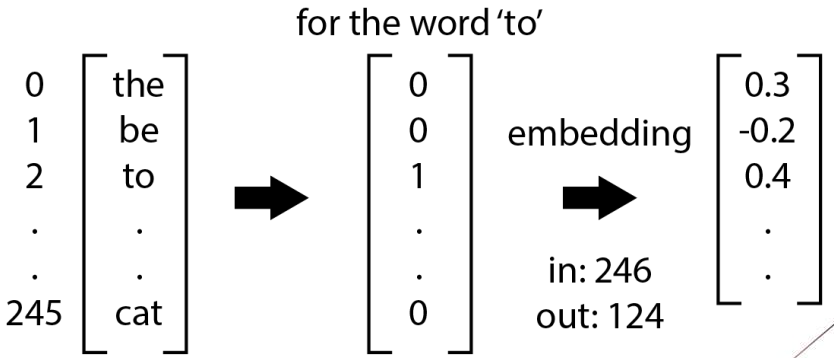
Set Random Seed for Initial Weight

```
tensorflow::tf$random$set_seed(123)
```

Build model architecture

```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = num_words,
                  input_length = maxlen,
                  output_dim = 8
                  ) %>%
  layer_lstm(units = 8,
             kernel_regularizer = regularizer_l1_l2(l1 = 0.05, l2 = 0.05),
             return_sequences = F
             ) %>%
  layer_dense(units = 1, activation = "sigmoid")
```

Embedding layer will convert encoded word into new vector



algoritma
data science education center

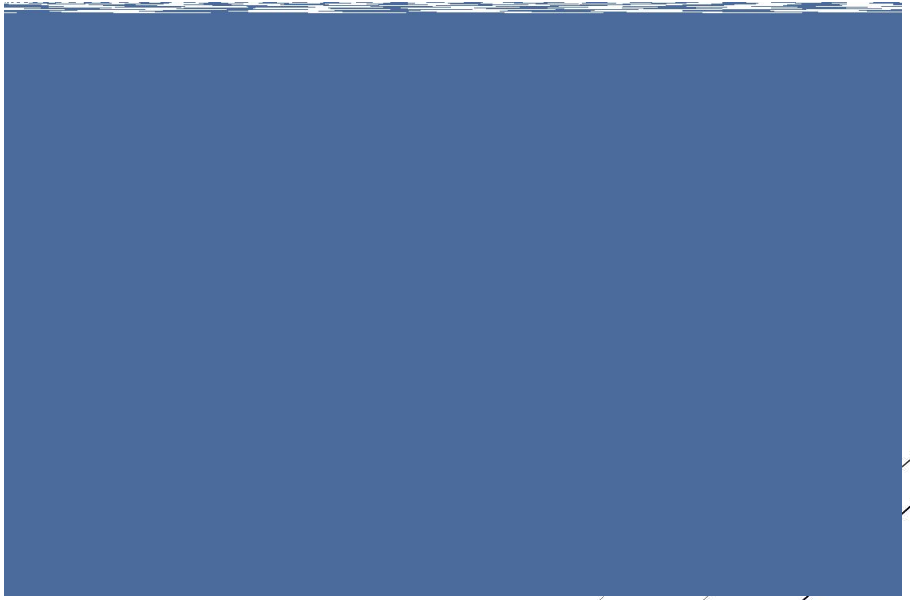
Model Architecture

Set Random Seed for Initial Weight

```
tensorflow::tf$random$set_seed(123)
```

Build model architecture

```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = num_words,
                  input_length = maxlen,
                  output_dim = 8
                  ) %>%
  layer_lstm(units = 8,
             kernel_regularizer = regularizer_l1_l2(l1 = 0.05, l2 = 0.05),
             return_sequences = F
             ) %>%
  layer_dense(units = 1, activation = "sigmoid")
```



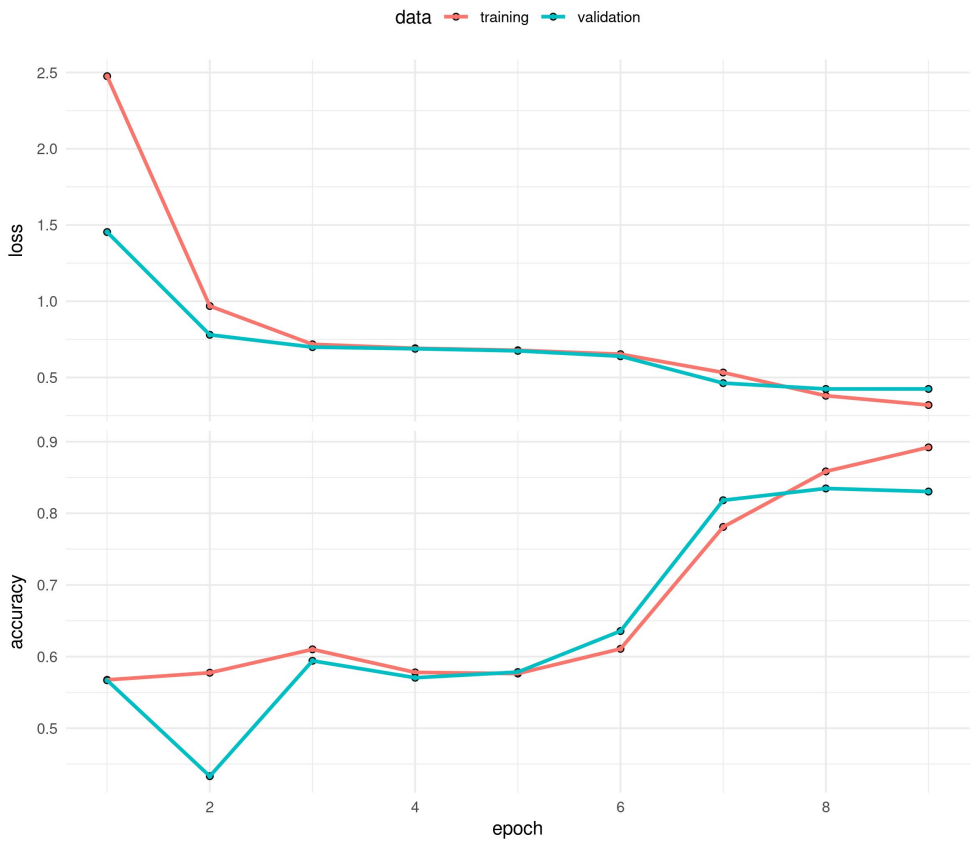
Train the Model

Building Model

```
model %>%
  compile(
        optimizer = optimizer_adam(lr = 0.001),
        metrics = "accuracy",
        loss = "binary_crossentropy"
        )
```

Model Fitting

```
train_history <- model %>%
  fit(x = train_x,
      y = train_y,
      batch_size = 64,
      epochs = 9,
      validation_split = 0.1
      )
```



Model Evaluation

Predict the Data Test

```
pred_test <- predict_classes(model, test_x)
```

Convert Target Variable into Interpretable Labels

```
decode <- function(x) as.factor(ifelse(x == 0, "Not Recommended", "Recommended"))

pred_class <- decode(pred_test)
true_class <- decode(data_test$user_suggestion)
```

Create Confusion Matrix

```
table("Prediction" = pred_class, "Actual" = true_class)
```

Result of Confusion Matrix

		Actual	
		Recommend	Not Recommend
Prediction	Recommend	1681	263
	Not Recommend	322	1225

Model Evaluation

		Actual	
		Recommend	Not Recommend
Prediction	Recommend	True Positive	False Positive
	Not Recommend	False Negative	True Negative

		Actual	
		Recommend	Not Recommend
Prediction	Recommend	1681	263
	Not Recommend	322	1225

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = 2\,\frac{precision \times recall}{precision + recall}$$

Model Evaluation

Get Model Performance Metrics

```
data.frame(
            Accuracy = accuracy_vec(pred_class, true_class),
            Recall = sens_vec(pred_class, true_class),
            Precision = precision_vec(pred_class, true_class),
            F1 = f_meas_vec(pred_class, true_class)
            )
```

Model	Accuracy	Sensitivity (Recall)	Precision	F1 Score
Deep Learning - LSTM	83.24%	83.92%	86.47%	85.18%
Naive Bayes	54.28%	68.75%	58.67%	63.31%

Feel Free to Contact me

arga@algorit.ma

github.com/Argaadya

Arga Adyatama

algoritma
data science education center

Your Feedback Matters!

bit.ly/3hmJ3Nr