Sona Text Analysis

Emma Stavrakis, Sarah Mbaka, Jaco de Swardt

20 September 2018

# 1. Introduction

In this assignment we will be conducting both predictive and descriptive analyses on the SONA dataset. To this end we have:

1. Built a neural network and a convolutional neural network to predict which president was the source when given a sentence of text
2. Assessed the performance of both the nn and cnn on unseen data and then compared and interpreted these results
3. Conducted a desriptive analysis of the text in the speeches

This report sets out the approach we took to conduct these analyses, decisions and choices that we made, what challenges we encountered and how we overcame these, as well as our interpretation of- and commentary on the results.

We were tasked to work in groups for this assignment and to advise how the workload was divided amongst the group members, as well as providing evidence of team collaboration on Git. We have covered this section at the end of this workbook. Our repo for this project may be accessed here: <https://github.com/emmastavrakis/DS-FI-Sona/>

# 2. Data preparation

The State of the Nation Address of the President of South Africa (SONA) is an annual event in which the President of South Africa reports on the status of the nation, normally to the resumption of a joint sitting of Parliament. In years that elections took place, a State of the Nation Address happens twice, once before and again after the election.

We were provided with full text of all State of the Nation Address (SONA) speeches, from 1994 through to 2018 on which to conduct our analyses.

## 2.1 Data challenges

The first challenge that we encountered was that some the data that was provided had inadvertantly been duplicated on kaggle (Mbeki’s speech from 2007 and 2008 were found to be the same). We replaced the incorrect data with the correct version directly in the source data files for this project.

The next challenge that we encountered is that some of the data categories were very sparse, as some presidents had given many more SONA speeches than others. Particularly, Presidents de Klerk, Mothlante and Ramaphosa had each only delivered one speech. We debated reducing the sample size of the other presidents accordingly, but this would have resulted in ignoring 75% of the total dataset size, which we decided againt. We debated over-sampling the sparse presidents, but some of the other groups advised that they had tried this already with very little impact on overall prediction accuracy. So we decided to leave the data as-is, and rather to focus on tuning our models.

## 2.2 Pre-processing

Our approach to pre-processing the data centred primarily on using the **tidyverse** and **tidytext** libraries to wrangle the data and mine the text.

We started by extracting metadata to identify the president who was the source of each speech, as well as the year that the speech was given. Initially we labelled the column with the president’s name as **“president”**, which caused problems later on with tokenization as the word “president” is also a token. We then changed this column reference to **“which\_pres”**.

## Warning: package 'tidyverse' was built under R version 3.5.1

## Warning: package 'ggplot2' was built under R version 3.5.1

## Warning: package 'purrr' was built under R version 3.5.1

## Warning: package 'dplyr' was built under R version 3.5.1

## Warning: package 'stringr' was built under R version 3.5.1

Thereafter we tokenized the words and sentences, produced counts of each token and then spread the data from long format to wide format as required by the neural networks that we are building. Below is a sample of the wrangled, tokenized, counted data, ready to be processed by the neural network.

## Warning: package 'tidytext' was built under R version 3.5.1

## # A tibble: 6 x 10,310  
## # Groups: ID, which\_pres [6]  
## ID which\_pres `0` `0,2` `0.5` `000` `01` `019` `03` `09` `1`  
## <int> <chr> <int> <int> <int> <int> <int> <int> <int> <int> <int>  
## 1 1 Mandela NA NA NA NA NA NA NA NA NA  
## 2 2 Mandela NA NA NA NA NA NA NA NA NA  
## 3 3 Mandela NA NA NA NA NA NA NA NA NA  
## 4 4 Mandela NA NA NA NA NA NA NA NA NA  
## 5 5 Mandela NA NA NA NA NA NA NA NA NA  
## 6 6 Mandela NA NA NA NA NA NA NA NA NA  
## # ... with 10,299 more variables: `1,000` <int>, `1,1` <int>, `1,2` <int>,  
## # `1,3` <int>, `1,300` <int>, `1,4` <int>, `1,6` <int>, `1,700` <int>,  
## # `1,9` <int>, `1.1` <int>, `1.2` <int>, `1.3` <int>, `1.4` <int>,  
## # `1.8` <int>, `10` <int>, `10,000` <int>, `10.4` <int>, `10.7` <int>,  
## # `100` <int>, `1000` <int>, `100th` <int>, `102` <int>, `105` <int>,  
## # `107` <int>, `108` <int>, `10th` <int>, `11` <int>, `11,2` <int>,  
## # `113` <int>, `11th` <int>, `12` <int>, `12,4` <int>, `12.1` <int>,  
## # `12.5` <int>, `120` <int>, `123rd` <int>, `124` <int>, `125` <int>,  
## # `126` <int>, `127` <int>, `129` <int>, `12th` <int>, `13` <int>,  
## # `130` <int>, `133` <int>, `136` <int>, `13th` <int>, `14` <int>,  
## # `14.5` <int>, `140` <int>, `145` <int>, `146` <int>, `149` <int>,  
## # `14th` <int>, `15` <int>, `15,000` <int>, `15,3` <int>, `15,8` <int>,  
## # `150` <int>, `150,000` <int>, `150th` <int>, `152` <int>, `155` <int>,  
## # `159` <int>, `15th` <int>, `16` <int>, `160` <int>, `168` <int>,  
## # `169` <int>, `16th` <int>, `17` <int>, `17.6` <int>, `170` <int>,  
## # `172` <int>, `175` <int>, `17th` <int>, `18` <int>, `18,4` <int>,  
## # `180` <int>, `187` <int>, `1879` <int>, `18th` <int>, `19` <int>,  
## # `1906` <int>, `1909` <int>, `191` <int>, `1910` <int>, `1912` <int>,  
## # `1913` <int>, `1917` <int>, `1919` <int>, `1940` <int>, `1955` <int>,  
## # `1956` <int>, `1957` <int>, `1960` <int>, `1960s` <int>, `1963` <int>,  
## # `1970s` <int>, `1974` <int>, ...

## 2.3 Data partitioning

In order to assess the performance of the predictive models, we partitioned the data into a training dataset and a validation dataset. We then extracted the response variable, **“which\_pres”** into its own dataframe and “one-hot” encoded it. Below is an extract of the one-hot encoded response variable from the training dataset.

## Warning: package 'listarrays' was built under R version 3.5.1

## Motlanthe deKlerk Mandela Mbeki Ramaphosa Zuma  
## [1,] 0 0 0 1 0 0  
## [2,] 0 0 0 0 0 1  
## [3,] 0 0 0 1 0 0  
## [4,] 0 0 0 0 0 1  
## [5,] 0 0 0 0 0 1  
## [6,] 0 0 1 0 0 0

# 3. Predictive Anaysis

For the predictive analysis we decided to build two models and compare their performance. We started with a basic neural network (nn) and then built a convolutional neural network (cnn).

## 3.1 Neural Network

The **keras** library was used to build the neural network.

## Warning: package 'keras' was built under R version 3.5.1

### 3.1.1 Define the nn model

We defined the nn model with the following architecture:

nn\_model %>%   
 layer\_dense(units = 256, activation = 'relu', input\_shape = c(ncols-1)) %>%   
 layer\_dropout(rate = 0.4) %>%   
 layer\_dense(units = 128, activation = 'relu') %>%  
 layer\_dropout(rate = 0.3) %>%  
 layer\_dense(units = 6, activation = 'softmax')

which is represented by the following summary:

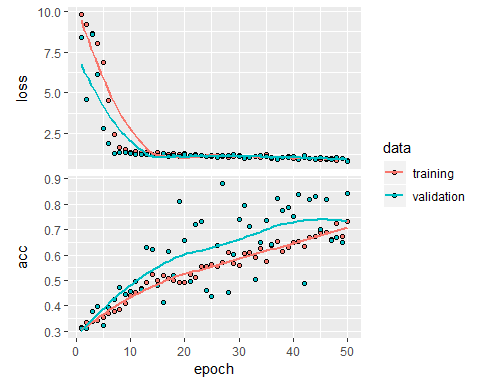
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## Layer (type) Output Shape Param #   
## ===========================================================================  
## dense\_1 (Dense) (None, 256) 2639360   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dropout\_1 (Dropout) (None, 256) 0   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense\_2 (Dense) (None, 128) 32896   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dropout\_2 (Dropout) (None, 128) 0   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense\_3 (Dense) (None, 6) 774   
## ===========================================================================  
## Total params: 2,673,030  
## Trainable params: 2,673,030  
## Non-trainable params: 0  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 3.1.2 Compile the model

Before the neural network can be trained, it first needs to be compipled. In this step we specified the loss type, optimisation algorithm and measurement metric. Because our predictive model has more than two possible categories to be predicted, we used categorical cross-entropy for the loss function. There are a number of optimizer algorithms that can be used, but for the purposes of a basic neural network we elected to use the **RMSProp** optimizer.

### 3.1.3 Train the model

In training the model, it is necessary to specify the epochs, batch size and proportion of data to be used for validation. We elected to use 20% of the data for validation, and then tuned the other hyper parameters manually to maximise the validation accuracy of the model. We found that values of 50 epochs and a batch size of 400 gave optimal results.



### 3.1.4 Evaluate the model on the unseen data

## $loss  
## [1] 0.8089562  
##   
## $acc  
## [1] 0.8329897

### 3.1.5 Cross validation

## Warning: package 'caret' was built under R version 3.5.1

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

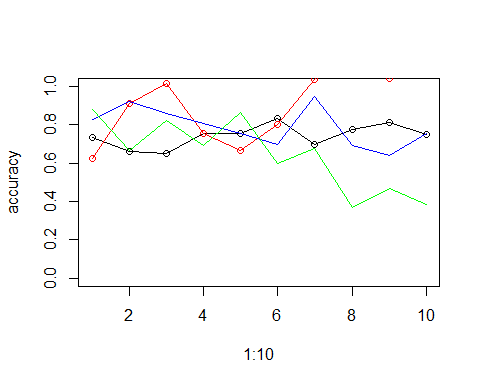
## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

## Warning in loss[i] <- crossval\_history$metrics$loss: number of items to  
## replace is not a multiple of replacement length

## Warning in accuracy[i] <- crossval\_history$metrics$acc: number of items to  
## replace is not a multiple of replacement length

## Warning in val\_loss[i] <- crossval\_history$metrics$val\_loss: number of  
## items to replace is not a multiple of replacement length

## Warning in val\_acc[i] <- crossval\_history$metrics$val\_acc: number of items  
## to replace is not a multiple of replacement length

 ###3.1.6 Generate predictions on new data