Predictive Modelling of Presidential SONA Addresses in South Africa - 1994 to 2023

Steven Ellis*

E-mail: ELLSTE005@uct.ac.za

Abstract

Summary of motivation and outcome. Start with context, task and object, finish with findings and conclusion. This is written last.

Introduction

The State of the Nation Address of the President of South Africa is an annual event in the Republic of South Africa, in which the President of South Africa reports on the status of the nation, normally to the resumption of a joint sitting of Parliament. This assignment purports to create predictive models that predict from a sentence from which South African president it derives. The data-set provided includes 36 speeches from 1994 to 2023 delivered by six different presidents.

For the assignment, the prodictive models generated and compared were a feed-forward neural network, a classification tree, a random forest and a convolutional neural network.

Literature Review

Data and Methods

Data import and transformation

As part of the data import and transformation step, the following tasks were performed: * All speeches were imported from their respective `.txt` files into a data-frame * The speech year and president were added as columns to the data-frame * Any links or non-ascii words were removed from the sentences via the **str_replace_all()** function from the R `stringr` library * The data-frame was converted to a tibble

Imbalanced Data

The first data problem to be confronted was that of imbalanced data. Two presidents (deKlerk and Motlanthe) only produced one SONA speech, meaning their words and sentences were extremely under-represented in the data-set. In addition, some presidents gave longer speeches with more sentences than others. Since imbalanced data can have an adverse effect on predictive modelling, it was decided that three different data-sets would be produced:

1. An *imbalanced* data-set, where presidential speeches were left in their original proportions
2. A *balanced* data-set: in this data-set, the speeches severely under-represented presidents (*deKlerk* and *Motlanthe*) were removed. On the remaining data-set, **under-sampling** was performed by limiting each president's sentences to the *minimum* number of sentences delivered by a president within the remaining consort (1665)
3. An *oversampled* data-set. Over-sampling involves adding more samples from under-represented classes. To achieve this, the **oversample_smote()** function by the R `scutr` library was used to create synthetic samples from under-represented presidential

sentences, with the end result that each president had an equal number of speeches in the final oversampled data-set.

The three data-sets were then be used and compared during predictive modelling.

In order to prepare the data-set into one upon which various predictive models could be run, the data-set needed to be tokenised and transformed from 'long' to 'wide' format, where each word (and the frequency count of how often it is represented in a sentence delivered by a president) is represented as a numeric column in a sparse and wide data-set.

To achieve this, the data-set was first tokenised into sentences, using the **unnest_tokens()** function from the R `tidytext` library, which split the speeches column into sentence tokens, flattening the resulting data-set into one-sentence-per-row.

Thereafter, two methodologies were used to word-tokenize the remaining data-set: bag-of-words method and term frequency-inverse-document-frequency (tf-idf) model.

Bag of Words

In a bag-of-words model, a document is represented by the frequency counts of the words used therein via a simplified representation that ignores word order and grammar.

To achieve this, sentences had all stop words removed and were then tokenised per word. Thereafter, from the tidy word-tokenised data-set, the top **200** utilised words were extracted. Thereafter, using the sentence-tokenised data-set, number of times each of these words was used in each sentence was calculated. Finally, the resulting data-set was re-shaped using R **pivot_wider()** function, so that each sentence was in its own row, and each word in its own column. This wide, sparse and untidy data-set was now ready for predictive modelling.

The bag-of-words transformation was performed on all three data-sets (*imbalanced*, *balanced* and *oversampled*).

TF-IDF

What is clear from reading some of the SONA speeches is that certain words and phrases are habitually repeated by all presidents. In order to assist with predictive modelling, a common technique is to downweigh words in a term that are used frequently by all presidents (such as ‘deliver’, ‘budget’, ‘economy’, ‘invest’) and upweigh words that are relatively more frequently by a single president. This is what tf-idf aims to achieve. It calculates a inverse-document-frequency-weighted-term-frequencies score for each word, which is low for words commonly used in many documents, and higher for higher for words that are not used by many documents in a corpus. TF-IDF scores were retrieved using the **bind_tf_idf()** function from the R `tidytext` library.

The tf-idf transformation was also performed on all three data-sets (*imbalanced*, *balanced* and *oversampled*).

Test and training data-sets

Once Bag of Words and TF-IDF transformations were completed on the three data-sets, they were split into training and testing sets. The split chosen was a 70/30 split, where 70% of the observations were used to train predictive models and 30 were used for testing.

Neural Networks

Neural networks are a subset of machine learning and are at the heart of deep learning algorithms. These models are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer.

Each node, or artificial neuron, connects to another and has an associated weight and threshold. Weights are very much like the coefficients used in a regression equation.

The weighted inputs are summed and passed through an activation function (which simply maps the summed weighted inputs to the output of the neuron, typically using a non-linear function). If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

This topology allows the network to combine the inputs in more complex ways and thus to model highly non-linear data-sets.

For this problem, the R `keras` library was used, and the **keras_model_sequential()** function was used to create a linear stack of layers in our deep learning model. We used Keras to assemble layers into a fully-connected multi-layer perceptron.

- 1024 units in input layer, with *ReLu* activation function
- dropout layer with a rate of 0.1
- 128 units in hidden layer with *ReLu* activation function
- dropout layer with a rate of 0.2
- 6 units in output layer with *softmax* function

In order to prepare the data-set for multi-class classification, one hot encoding was performed on the target attribute target using the **to_categorical()** function in R `keras` library.

The **Adam** optimiser was used, and the **categorical_crossentropy** loss function was used because of the multi-class classification requirement, and the chosen metric was **accuracy**.

When fitting the model a batch_size of 128 was applied and 50 epochs were chosen. A validation split of 20% used.

Classification Tree

Classification trees are used for data classification through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches. Classification trees are also the fundamental components to random forests (used later), and allow for visualisation of decision rules and partitioning logic.

To generate classification trees models the **rpart()** R function was used.

Random Forest

While decision trees are common supervised learning algorithms, they can be prone to problems, such as bias and overfitting. Random forest strives to oversome this weakness by combining the output of multiple decision trees to reach a single result. Random forests ensure that the decision trees are *de-correlated* as follows: when building decision trees in a random forest, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. This prevents dominant predictors giving all trees in the model correlated results.

To generate random forest classification models, the **randomForest()** R function was used. A forest of 100 trees was generated.

Convolutional Neural Networks

Convolutional neural network is a regularized type of feed-forward neural network that learns feature engineering by itself via filters optimization. It is typically used for image recognition.

The pre-processing required in a convolutional neural network is much lower as compared to other classification algorithms. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and the reusability of weights.

To generate the CNN models, the **text_tokenizer()** and **fit_text_tokenizer()** functions from `keras` was used to vectorize the top X words from a balanced and unbalanced data-sets.

These were then padded to equal length before being fitted to a convolutional network via the **keras_model_sequential()** function. The exercise was repeated for top 100, 200 and 300 words.

One hot encoding was again performed on the target attribute target using the **to_categorical()** function in R `keras` library.

The **Adam** optimiser was used, and the **categorical_crossentropy** loss function was used, and the chosen metric was **accuracy**.

When fitting the model a batch_size of 128 was applied and 50 epochs were chosen. A validation split of 20% used.

Results

Classification Tree

Table 1: Table 1: Classification Tree Model Prediction Accuracies

Tokenisation model	Accuracy
Bag of words - unbalanced	0.314
Bag of words - balanced	0.294
Bag of words - oversampled	0.314
TFIDF - unbalanced	0.317
TFIDF - balanced	0.280
TFIDF - oversampled	0.309

From the results in **Table 1** it is clear that the classification tree model generally performed poorly against all data-sets, with (interestingly) slightly better prediction results against the unbalanced data-sets.

Table 2: Table 2: Neural Network Model Prediction Accuracies

neural_networks_results_df_type	neural_networks_results_accuracy
Bag of words - unbalanced	0.407
Bag of words - balanced	0.425
Bag of words - oversampled	0.645
TFIDF - unbalanced	0.433
TFIDF - balanced	0.417
TFIDF - oversampled	0.635

The feed-forward neural-network performed well against the over-sampled data-sets.

Table 3: Table 3: Neural Network - Confusion Matrix Against **Balanced Bag-of-Words** Data-Set

	Mandela	Mbeki	Ramaphosa	Zuma
Mandela	189	130	63	74
Mbeki	113	208	65	70
Ramaphosa	64	83	197	112
Zuma	82	87	106	181

Table 4: Table 4: Neural Network - Confusion Matrix Against **Balanced TFIDF** Data-Set

	Mandela	Mbeki	Ramaphosa	Zuma
Mandela	220	110	64	62
Mbeki	118	185	70	83
Ramaphosa	102	76	176	102
Zuma	85	87	104	180

Looking at the confusion matrix of the model against the balanced bag-of-words and tf-idf data-sets, whilst the diagonal (correct predictions) represents the highest number for each row (president), the model did suffer from a reasonably high portion of incorrect predictions.

Table 5: Table 5: Neural Network - Confusion Matrix Against **Oversampled TFIDF** Data-Set

	Mandela	Mbeki	Ramaphosa	Zuma	deKlerk	Motlanthe
Mandela	394	114	56	95	14	28
Mbeki	147	272	105	115	20	42
Ramaphosa	100	75	310	173	21	22
Zuma	78	91	151	346	21	14
deKlerk	7	0	1	4	689	0
Motlanthe	4	5	11	16	4	661

The confusion matrix for the over-sampled tf-idf data-set reveals an interesting phenomenon: the model achieved extremely high predictive accuracy against deKlerk and Motlanthe, both of whom where heavily over-sampled because of the fact that they delivered only one SONA speech. However, the remaining presidents also exhibit better prediction rates when compared to the two confusion matrices above, suggesting that the over-sampling method is effective in balancing out the data-set for predictive modelling.

Table 6: Table 6: Random Forest Model Prediction Accuracies

Tokenisation model	Accuracy
Bag of words - unbalanced	0.346
Bag of words - balanced	0.395
Bag of words - oversampled	0.481
TFIDF - unbalanced	0.348

Tokenisation model	Accuracy
TFIDF - balanced	0.375
TFIDF - oversampled	0.455

The random forest classifier produced satisfactory results across all data-sets, achieving over 48% accuracy against the over-sampled bag-of-words data-set.

Table 7: Table 7: Convolutional Neural Network Model Prediction Accuracies

Data Set	Loss	Accuracy
Top 100 words - unbalanced data set	1.349	0.442
Top 100 words - balanced data set	1.247	0.433
Top 200 words - unbalanced data set	1.289	0.477
Top 200 words - balanced data set	1.178	0.469
Top 300 words - unbalanced data set	1.240	0.489
Top 300 words - balanced data set	1.204	0.471

Finally, the CNN classifier also produced satisfactory results across all data-sets, achieving almost 49% accuracy against the top 300 tokenised words against an unbalanced data-set.

Discussion & Conclusion

References

The class makes various changes to the way that references are handled. The class loads `natbib`, and also the appropriate bibliography style. References can be made using the normal method; the citation should be placed before any punctuation, as the class will move it if using a superscript citation style[1]. The use of `natbib` allows the use of the various citation commands of that package have shown something. Long lists of authors will be automatically truncated in most article formats, but not in supplementary information or reviews. If you encounter problems with the citation macros, please check that your copy of `natbib` is up to date. The demonstration database file `bibliography.bib` shows how to complete entries correctly.

Multiple citations to be combined into a list can be given as a single citation. This uses the `mciteplus` package. Citations other than the first of the list should be indicated with a star.

The class also handles notes to be added to the bibliography. These should be given in place in the document. As with citations, the text should be placed before punctuation. A note is also generated if a citation has an optional note. This assumes that the whole work has already been cited: odd numbering will result if this is not the case .

References

References

(1) Garnier, S.; Gautrais, J.; Theraulaz, G. The biological principles of swarm intelligence. *Swarm Intelligence* **2007**, *1*, 3–31.